

artdaq
v3_00_00_RC2

Generated by Doxygen 1.8.5

Mon Dec 11 2017 10:51:23

Contents

1	Todo List	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	9
4.1	Class List	9
5	File Index	15
5.1	File List	15
6	Namespace Documentation	19
6.1	anonymous_namespace{genToArt.cc} Namespace Reference	19
6.1.1	Function Documentation	19
6.1.1.1	process_cmd_line	19
6.1.1.2	process_data	19
6.2	anonymous_namespace{rootOutputConfigurationTools.cc} Namespace Reference	20
6.2.1	Function Documentation	20
6.2.1.1	maxCriterionSpecified	20
6.3	anonymous_namespace{shared_memory_reader_t.cc} Namespace Reference	20
6.3.1	Function Documentation	21
6.3.1.1	basic_test	21
6.4	anonymous_namespace{TransferWrapper.cc} Namespace Reference	21
6.5	art Namespace Reference	21
6.5.1	Detailed Description	22
6.5.2	Function Documentation	22
6.5.2.1	printProcessHistoryID	22

6.5.2.2	printProcessMap	23
6.5.2.3	ReadObjectAny	23
6.6	artdaq Namespace Reference	23
6.6.1	Detailed Description	28
6.6.2	Typedef Documentation	28
6.6.2.1	makeFunc_t	28
6.6.3	Function Documentation	28
6.6.3.1	cmd_::getParam< art::RunID >	28
6.6.3.2	cmd_::getParam< fhicl::ParameterSet >	29
6.6.3.3	cmd_::getParam< std::string >	29
6.6.3.4	exception_msg	29
6.6.3.5	exception_msg	30
6.6.3.6	exception_msg	30
6.6.3.7	exception_msg	30
6.6.3.8	infoFilename	31
6.6.3.9	makeCommandableFragmentGenerator	32
6.6.3.10	MakeCommanderPlugin	32
6.6.3.11	makeRoutingMasterPolicy	32
6.6.3.12	MakeTransferPlugin	33
6.7	artdaq::detail Namespace Reference	33
6.7.1	Detailed Description	34
6.7.2	Enumeration Type Documentation	34
6.7.2.1	RequestMessageMode	34
6.7.2.2	RoutingMasterMode	35
6.7.3	Function Documentation	35
6.7.3.1	operator<<	35
7	Class Documentation	37
7.1	artdaq::art_config_file Class Reference	37
7.1.1	Detailed Description	37
7.1.2	Constructor & Destructor Documentation	37
7.1.2.1	art_config_file	37
7.1.3	Member Function Documentation	37
7.1.3.1	getFileName	38
7.2	art::ArtdaqInput< U > Class Template Reference	38
7.2.1	Detailed Description	38
7.2.2	Constructor & Destructor Documentation	39

7.2.2.1	ArtdaqInput	39
7.2.3	Member Function Documentation	39
7.2.3.1	hasMoreData	39
7.2.3.2	operator=	39
7.2.3.3	readFile	39
7.2.3.4	readNext	40
7.3	artdaq::AutodetectTransfer Class Reference	40
7.3.1	Detailed Description	41
7.3.2	Constructor & Destructor Documentation	41
7.3.2.1	AutodetectTransfer	41
7.3.3	Member Function Documentation	41
7.3.3.1	copyFragment	41
7.3.3.2	moveFragment	42
7.3.3.3	receiveFragment	43
7.3.3.4	receiveFragmentData	43
7.3.3.5	receiveFragmentHeader	43
7.4	art::BinaryFileOutput Class Reference	44
7.4.1	Detailed Description	44
7.4.2	Constructor & Destructor Documentation	44
7.4.2.1	BinaryFileOutput	44
7.5	art::BinaryMPIOutput Class Reference	45
7.5.1	Detailed Description	45
7.5.2	Constructor & Destructor Documentation	45
7.5.2.1	BinaryMPIOutput	45
7.6	artdaq::BoardReaderApp Class Reference	46
7.6.1	Detailed Description	47
7.6.2	Constructor & Destructor Documentation	47
7.6.2.1	BoardReaderApp	47
7.6.3	Member Function Documentation	47
7.6.3.1	BootedEnter	47
7.6.3.2	do_initialize	47
7.6.3.3	do_pause	48
7.6.3.4	do_reinitialize	48
7.6.3.5	do_resume	48
7.6.3.6	do_shutdown	49
7.6.3.7	do_soft_initialize	49
7.6.3.8	do_start	49

7.6.3.9	do_stop	50
7.6.3.10	operator=	50
7.6.3.11	report	50
7.7	artdaq::BoardReaderCore Class Reference	51
7.7.1	Detailed Description	52
7.7.2	Constructor & Destructor Documentation	52
7.7.2.1	BoardReaderCore	52
7.7.3	Member Function Documentation	52
7.7.3.1	GetDataSenderManagerPtr	52
7.7.3.2	initialize	52
7.7.3.3	operator=	53
7.7.3.4	pause	53
7.7.3.5	process_fragments	53
7.7.3.6	reinitialize	54
7.7.3.7	report	55
7.7.3.8	resume	55
7.7.3.9	shutdown	55
7.7.3.10	soft_initialize	55
7.7.3.11	start	56
7.7.3.12	stop	56
7.8	Builder Class Reference	56
7.8.1	Detailed Description	57
7.8.2	Constructor & Destructor Documentation	57
7.8.2.1	Builder	57
7.9	artdaq::BuildInfo< instanceName, Pkgs > Class Template Reference	57
7.9.1	Detailed Description	58
7.9.2	Constructor & Destructor Documentation	58
7.9.2.1	BuildInfo	58
7.9.3	Member Function Documentation	58
7.9.3.1	beginRun	58
7.9.3.2	produce	59
7.10	artdaq::CapacityTestPolicy Class Reference	59
7.10.1	Detailed Description	59
7.10.2	Constructor & Destructor Documentation	60
7.10.2.1	CapacityTestPolicy	60
7.10.3	Member Function Documentation	60
7.10.3.1	GetCurrentTable	60

7.11 artdaq::cmd_ Class Reference	60
7.11.1 Detailed Description	62
7.11.2 Constructor & Destructor Documentation	62
7.11.2.1 cmd_	62
7.11.3 Member Function Documentation	62
7.11.3.1 execute	62
7.11.3.2 execute_	62
7.11.3.3 getParam	63
7.11.3.4 getParam	63
7.11.3.5 getParam	64
7.12 artdaq::Commandable Class Reference	64
7.12.1 Detailed Description	66
7.12.2 Constructor & Destructor Documentation	66
7.12.2.1 Commandable	66
7.12.3 Member Function Documentation	66
7.12.3.1 badTransition	66
7.12.3.2 BootedEnter	66
7.12.3.3 current_state	67
7.12.3.4 do_initialize	67
7.12.3.5 do_pause	67
7.12.3.6 do_reinitialize	67
7.12.3.7 do_resume	68
7.12.3.8 do_shutdown	68
7.12.3.9 do_soft_initialize	68
7.12.3.10 do_start	68
7.12.3.11 do_stop	69
7.12.3.12 in_run_failure	69
7.12.3.13 initialize	69
7.12.3.14 InRunExit	69
7.12.3.15 legal_commands	69
7.12.3.16 operator=	70
7.12.3.17 pause	70
7.12.3.18 register_monitor	70
7.12.3.19 reinitialize	70
7.12.3.20 report	71
7.12.3.21 resume	71
7.12.3.22 shutdown	71

7.12.3.23 soft_initialize	71
7.12.3.24 start	72
7.12.3.25 status	72
7.12.3.26 stop	72
7.12.3.27 unregister_monitor	73
7.13 artdaq::CommandableFragmentGenerator Class Reference	73
7.13.1 Detailed Description	76
7.13.2 Constructor & Destructor Documentation	76
7.13.2.1 CommandableFragmentGenerator	76
7.13.2.2 CommandableFragmentGenerator	76
7.13.2.3 ~CommandableFragmentGenerator	77
7.13.3 Member Function Documentation	77
7.13.3.1 applyRequests	77
7.13.3.2 applyRequestsBufferMode	77
7.13.3.3 applyRequestsIgnoredMode	78
7.13.3.4 applyRequestsSingleMode	79
7.13.3.5 applyRequestsWindowMode	79
7.13.3.6 board_id	79
7.13.3.7 check_stop	79
7.13.3.8 dataBufferIsTooLarge	79
7.13.3.9 ev_counter	80
7.13.3.10 ev_counter_inc	80
7.13.3.11 exception	80
7.13.3.12 fragment_id	80
7.13.3.13 fragmentIDs	80
7.13.3.14 getDataBufferStats	81
7.13.3.15 getNext	81
7.13.3.16 metricsReportingInstanceName	81
7.13.3.17 metricsReportingInstanceName	81
7.13.3.18 PauseCmd	81
7.13.3.19 printMode_	82
7.13.3.20 ReportCmd	82
7.13.3.21 ResumeCmd	82
7.13.3.22 run_number	82
7.13.3.23 sendEmptyFragment	83
7.13.3.24 sendEmptyFragments	83
7.13.3.25 set_exception	83

7.13.3.26 <code>should_stop</code>	83
7.13.3.27 <code>StartCmd</code>	83
7.13.3.28 <code>StopCmd</code>	84
7.13.3.29 <code>subrun_number</code>	84
7.13.3.30 <code>timeout</code>	84
7.13.3.31 <code>timestamp</code>	84
7.13.3.32 <code>waitForDataBufferReady</code>	85
7.14 <code>artdaqtest::CommandableFragmentGeneratorTest</code> Class Reference	85
7.14.1 Detailed Description	86
7.14.2 Member Function Documentation	86
7.14.2.1 <code>checkHWStatus_</code>	86
7.14.2.2 <code>getNext_</code>	86
7.14.2.3 <code>setFireCount</code>	86
7.14.2.4 <code>waitForFrags</code>	87
7.15 <code>artdaq::CommanderInterface</code> Class Reference	87
7.15.1 Detailed Description	88
7.15.2 Constructor & Destructor Documentation	88
7.15.2.1 <code>CommanderInterface</code>	88
7.15.3 Member Function Documentation	89
7.15.3.1 <code>operator=</code>	89
7.15.3.2 <code>run_server</code>	89
7.15.3.3 <code>send_init</code>	89
7.15.3.4 <code>send_legal_commands</code>	89
7.15.3.5 <code>send_pause</code>	90
7.15.3.6 <code>send_register_monitor</code>	90
7.15.3.7 <code>send_reinit</code>	90
7.15.3.8 <code>send_report</code>	90
7.15.3.9 <code>send_resume</code>	91
7.15.3.10 <code>send_shutdown</code>	91
7.15.3.11 <code>send_soft_init</code>	91
7.15.3.12 <code>send_start</code>	91
7.15.3.13 <code>send_status</code>	92
7.15.3.14 <code>send_stop</code>	92
7.15.3.15 <code>send_unregister_monitor</code>	92
7.15.4 Member Data Documentation	92
7.15.4.1 <code>_commandable</code>	92
7.16 <code>artdaq::CompositeDriver</code> Class Reference	93

7.16.1	Detailed Description	93
7.16.2	Constructor & Destructor Documentation	93
7.16.2.1	CompositeDriver	93
7.17	artdaq::DataLoggerApp Class Reference	94
7.17.1	Detailed Description	95
7.17.2	Constructor & Destructor Documentation	95
7.17.2.1	DataLoggerApp	95
7.17.3	Member Function Documentation	95
7.17.3.1	do_initialize	95
7.17.3.2	do_pause	95
7.17.3.3	do_reinitialize	96
7.17.3.4	do_resume	96
7.17.3.5	do_shutdown	96
7.17.3.6	do_soft_initialize	96
7.17.3.7	do_start	97
7.17.3.8	do_stop	98
7.17.3.9	operator=	98
7.17.3.10	report	98
7.18	artdaq::DataLoggerCore Class Reference	99
7.18.1	Detailed Description	99
7.18.2	Constructor & Destructor Documentation	99
7.18.2.1	DataLoggerCore	99
7.18.2.2	~DataLoggerCore	100
7.18.3	Member Function Documentation	100
7.18.3.1	initialize	100
7.18.3.2	operator=	100
7.19	artdaq::DataReceiverCore Class Reference	101
7.19.1	Detailed Description	102
7.19.2	Constructor & Destructor Documentation	102
7.19.2.1	DataReceiverCore	102
7.19.2.2	~DataReceiverCore	103
7.19.3	Member Function Documentation	103
7.19.3.1	initialize	103
7.19.3.2	initializeDataReceiver	103
7.19.3.3	logMessage_	104
7.19.3.4	operator=	105
7.19.3.5	pause	105

7.19.3.6	reinitialize	105
7.19.3.7	report	105
7.19.3.8	resume	106
7.19.3.9	shutdown	106
7.19.3.10	soft_initialize	106
7.19.3.11	start	106
7.19.3.12	stop	106
7.20	artdaq::DataReceiverManager Class Reference	107
7.20.1	Detailed Description	107
7.20.2	Constructor & Destructor Documentation	108
7.20.2.1	DataReceiverManager	108
7.20.3	Member Function Documentation	108
7.20.3.1	byteCount	108
7.20.3.2	count	108
7.20.3.3	enabled_sources	108
7.20.3.4	GetReceivedFragmentCount	109
7.20.3.5	getSharedMemoryEventManager	109
7.20.3.6	running_sources	109
7.20.3.7	slotCount	109
7.21	artdaq::DataSenderManager Class Reference	109
7.21.1	Detailed Description	110
7.21.2	Constructor & Destructor Documentation	110
7.21.2.1	DataSenderManager	110
7.21.3	Member Function Documentation	111
7.21.3.1	count	111
7.21.3.2	destinationCount	111
7.21.3.3	enabled_destinations	111
7.21.3.4	GetRoutingTableEntryCount	111
7.21.3.5	sendFragment	111
7.21.3.6	slotCount	112
7.22	artdaq::DispatcherApp Class Reference	112
7.22.1	Detailed Description	113
7.22.2	Constructor & Destructor Documentation	113
7.22.2.1	DispatcherApp	113
7.22.3	Member Function Documentation	113
7.22.3.1	do_initialize	113
7.22.3.2	do_pause	114

7.22.3.3 <code>do_reinitialize</code>	114
7.22.3.4 <code>do_resume</code>	114
7.22.3.5 <code>do_shutdown</code>	114
7.22.3.6 <code>do_soft_initialize</code>	115
7.22.3.7 <code>do_start</code>	115
7.22.3.8 <code>do_stop</code>	115
7.22.3.9 <code>operator=</code>	115
7.22.3.10 <code>register_monitor</code>	116
7.22.3.11 <code>report</code>	117
7.22.3.12 <code>unregister_monitor</code>	117
7.23 <code>artdaq::DispatcherCore</code> Class Reference	117
7.23.1 Detailed Description	118
7.23.2 Constructor & Destructor Documentation	118
7.23.2.1 <code>DispatcherCore</code>	118
7.23.2.2 <code>~DispatcherCore</code>	119
7.23.3 Member Function Documentation	119
7.23.3.1 <code>initialize</code>	119
7.23.3.2 <code>operator=</code>	119
7.23.3.3 <code>register_monitor</code>	120
7.23.3.4 <code>unregister_monitor</code>	121
7.24 <code>art::detail::DummyProductCache</code> Class Reference	121
7.24.1 Detailed Description	121
7.24.2 Member Function Documentation	121
7.24.2.1 <code>product</code>	121
7.25 <code>artdaq::EventBuilderApp</code> Class Reference	122
7.25.1 Detailed Description	123
7.25.2 Constructor & Destructor Documentation	123
7.25.2.1 <code>EventBuilderApp</code>	123
7.25.3 Member Function Documentation	123
7.25.3.1 <code>BootedEnter</code>	123
7.25.3.2 <code>do_initialize</code>	123
7.25.3.3 <code>do_pause</code>	124
7.25.3.4 <code>do_reinitialize</code>	124
7.25.3.5 <code>do_resume</code>	124
7.25.3.6 <code>do_shutdown</code>	125
7.25.3.7 <code>do_soft_initialize</code>	125
7.25.3.8 <code>do_start</code>	125

7.25.3.9 <code>do_stop</code>	125
7.25.3.10 <code>operator=</code>	126
7.25.3.11 <code>report</code>	126
7.26 <code>artdaq::EventBuilderCore</code> Class Reference	126
7.26.1 Detailed Description	127
7.26.2 Constructor & Destructor Documentation	127
7.26.2.1 <code>EventBuilderCore</code>	127
7.26.2.2 <code>~EventBuilderCore</code>	127
7.26.3 Member Function Documentation	127
7.26.3.1 <code>initialize</code>	127
7.26.3.2 <code>operator=</code>	128
7.27 <code>artdaq::EventDump</code> Class Reference	128
7.27.1 Detailed Description	128
7.27.2 Constructor & Destructor Documentation	129
7.27.2.1 <code>EventDump</code>	129
7.27.3 Member Function Documentation	130
7.27.3.1 <code>analyze</code>	130
7.28 <code>artdaq::detail::FragCounter</code> Class Reference	130
7.28.1 Detailed Description	131
7.28.2 Member Function Documentation	131
7.28.2.1 <code>count</code>	131
7.28.2.2 <code>incSlot</code>	131
7.28.2.3 <code>incSlot</code>	131
7.28.2.4 <code>minCount</code>	131
7.28.2.5 <code>nSlots</code>	131
7.28.2.6 <code>operator[]</code>	132
7.28.2.7 <code>setSlot</code>	132
7.28.2.8 <code>slotCount</code>	132
7.29 <code>artdaq::FragmentReceiverManager</code> Class Reference	132
7.29.1 Detailed Description	133
7.29.2 Constructor & Destructor Documentation	133
7.29.2.1 <code>FragmentReceiverManager</code>	133
7.29.3 Member Function Documentation	134
7.29.3.1 <code>byteCount</code>	134
7.29.3.2 <code>count</code>	134
7.29.3.3 <code>enabled_sources</code>	134
7.29.3.4 <code>recvFragment</code>	134

7.29.3.5	slotCount	134
7.30	artdaq::FragmentSniffer Class Reference	135
7.30.1	Detailed Description	135
7.30.2	Constructor & Destructor Documentation	136
7.30.2.1	FragmentSniffer	136
7.30.3	Member Function Documentation	137
7.30.3.1	analyze	137
7.31	artdaq::FragmentStoreElement Class Reference	137
7.31.1	Detailed Description	138
7.31.2	Member Function Documentation	138
7.31.2.1	emplace_back	138
7.31.2.2	emplace_front	138
7.31.2.3	empty	138
7.31.2.4	front	138
7.31.2.5	GetEndOfData	139
7.31.2.6	SetEndOfData	139
7.32	artdaq::GenericFragmentSimulator Class Reference	139
7.32.1	Detailed Description	140
7.32.2	Member Enumeration Documentation	140
7.32.2.1	content_selector_t	140
7.32.3	Constructor & Destructor Documentation	140
7.32.3.1	GenericFragmentSimulator	140
7.32.4	Member Function Documentation	141
7.32.4.1	fragmentIDs	141
7.32.4.2	getNext	141
7.32.4.3	getNext	141
7.33	artdaq::GetPackageBuildInfo Struct Reference	142
7.33.1	Detailed Description	142
7.33.2	Member Function Documentation	142
7.33.2.1	getPackageBuildInfo	142
7.34	artdaq::Globals Class Reference	142
7.34.1	Detailed Description	143
7.34.2	Member Function Documentation	143
7.34.2.1	seedAndRandom_	143
7.34.2.2	timevalAsDouble	143
7.35	artdaq::init_ Class Reference	143
7.35.1	Detailed Description	144

7.35.2 Constructor & Destructor Documentation	144
7.35.2.1 init_	144
7.35.3 Member Data Documentation	144
7.35.3.1 defaultTimeout	144
7.35.3.2 defaultTimestamp	144
7.36 artdaq::legal_commands_Class Reference	145
7.36.1 Detailed Description	145
7.36.2 Constructor & Destructor Documentation	145
7.36.2.1 legal_commands_	145
7.37 LockFile Class Reference	145
7.37.1 Detailed Description	146
7.37.2 Constructor & Destructor Documentation	146
7.37.2.1 LockFile	146
7.37.3 Member Function Documentation	146
7.37.3.1 IsLocked	146
7.38 MessHead Struct Reference	147
7.38.1 Detailed Description	147
7.38.2 Member Enumeration Documentation	147
7.38.2.1 MessType	147
7.39 MPIProg Struct Reference	148
7.39.1 Detailed Description	148
7.39.2 Constructor & Destructor Documentation	148
7.39.2.1 MPIProg	148
7.40 artdaq::MPISentry Class Reference	149
7.40.1 Detailed Description	149
7.40.2 Constructor & Destructor Documentation	149
7.40.2.1 MPISentry	149
7.40.2.2 MPISentry	149
7.40.2.3 MPISentry	150
7.40.3 Member Function Documentation	150
7.40.3.1 procs	150
7.40.3.2 rank	150
7.40.3.3 threading_level	150
7.41 artdaq::MPITransfer Class Reference	151
7.41.1 Detailed Description	151
7.41.2 Constructor & Destructor Documentation	152
7.41.2.1 MPITransfer	152

7.41.3 Member Function Documentation	152
7.41.3.1 copyFragment	152
7.41.3.2 moveFragment	152
7.41.3.3 operator=	153
7.41.3.4 receiveFragmentData	153
7.41.3.5 receiveFragmentHeader	153
7.42 MPRGlobalTestFixture Class Reference	153
7.42.1 Detailed Description	154
7.42.2 Member Function Documentation	154
7.42.2.1 fake_single_module_process	154
7.42.2.2 fake_single_process_branch	155
7.43 artdaq::MulticastTransfer Class Reference	155
7.43.1 Detailed Description	156
7.43.2 Constructor & Destructor Documentation	156
7.43.2.1 MulticastTransfer	156
7.43.3 Member Function Documentation	157
7.43.3.1 copyFragment	157
7.43.3.2 moveFragment	158
7.43.3.3 receiveFragment	158
7.43.3.4 receiveFragmentData	158
7.43.3.5 receiveFragmentHeader	159
7.44 artdaq::NetMonHeader Struct Reference	159
7.44.1 Detailed Description	159
7.45 NetMonTransportService Class Reference	160
7.45.1 Detailed Description	160
7.45.2 Constructor & Destructor Documentation	160
7.45.2.1 NetMonTransportService	160
7.45.3 Member Function Documentation	161
7.45.3.1 connect	161
7.45.3.2 dataReceiverCount	161
7.45.3.3 disconnect	161
7.45.3.4 receiveInitMessage	161
7.45.3.5 receiveMessage	161
7.45.3.6 sendMessage	162
7.46 NetMonTransportServiceInterface Class Reference	162
7.46.1 Detailed Description	163
7.46.2 Member Function Documentation	163

7.46.2.1	connect	163
7.46.2.2	disconnect	163
7.46.2.3	listen	163
7.46.2.4	receiveInitMessage	163
7.46.2.5	receiveMessage	163
7.46.2.6	sendMessage	164
7.47	art::NetMonWrapper Class Reference	164
7.47.1	Detailed Description	164
7.47.2	Constructor & Destructor Documentation	165
7.47.2.1	NetMonWrapper	165
7.47.3	Member Function Documentation	165
7.47.3.1	receiveInitMessage	165
7.47.3.2	receiveMessage	165
7.48	artdaq::NoOpPolicy Class Reference	165
7.48.1	Detailed Description	166
7.48.2	Constructor & Destructor Documentation	166
7.48.2.1	NoOpPolicy	166
7.48.3	Member Function Documentation	166
7.48.3.1	GetCurrentTable	166
7.49	artdaq::NullTransfer Class Reference	166
7.49.1	Detailed Description	167
7.49.2	Constructor & Destructor Documentation	167
7.49.2.1	NullTransfer	167
7.49.3	Member Function Documentation	168
7.49.3.1	copyFragment	168
7.49.3.2	moveFragment	168
7.49.3.3	receiveFragment	168
7.49.3.4	receiveFragmentData	168
7.49.3.5	receiveFragmentHeader	169
7.50	artdaq::RTIDDS::OctetsListener Class Reference	169
7.50.1	Detailed Description	169
7.50.2	Member Function Documentation	169
7.50.2.1	on_data_available	169
7.50.2.2	receiveFragmentFromDDS	170
7.51	artdaq::pause_ Class Reference	170
7.51.1	Detailed Description	171
7.51.2	Constructor & Destructor Documentation	171

7.51.2.1 pause_	171
7.51.3 Member Data Documentation	171
7.51.3.1 defaultTimeout	171
7.51.3.2 defaultTimestamp	171
7.52 artdaq::PrintBuildInfo Class Reference	171
7.52.1 Detailed Description	172
7.52.2 Constructor & Destructor Documentation	172
7.52.2.1 PrintBuildInfo	172
7.52.3 Member Function Documentation	172
7.52.3.1 beginRun	172
7.53 artdaq::RandomDelayFilter Class Reference	173
7.53.1 Detailed Description	173
7.53.2 Constructor & Destructor Documentation	173
7.53.2.1 RandomDelayFilter	173
7.53.3 Member Function Documentation	174
7.53.3.1 filter	174
7.53.3.2 operator=	174
7.53.3.3 operator=	174
7.53.3.4 reconfigure	174
7.54 daqrate.Re Class Reference	175
7.54.1 Detailed Description	175
7.55 artdaq::register_monitor_ Class Reference	175
7.55.1 Detailed Description	176
7.55.2 Constructor & Destructor Documentation	176
7.55.2.1 register_monitor_	176
7.56 artdaq::reinit_ Class Reference	176
7.56.1 Detailed Description	177
7.56.2 Constructor & Destructor Documentation	177
7.56.2.1 reinit_	177
7.56.3 Member Data Documentation	177
7.56.3.1 defaultTimeout	177
7.56.3.2 defaultTimestamp	177
7.57 artdaq::report_ Class Reference	177
7.57.1 Detailed Description	178
7.57.2 Constructor & Destructor Documentation	178
7.57.2.1 report_	178
7.58 artdaq::detail::RequestHeader Struct Reference	178

7.58.1	Detailed Description	179
7.58.2	Member Function Documentation	179
7.58.2.1	isValid	179
7.58.3	Member Data Documentation	179
7.58.3.1	header	179
7.59	artdaq::detail::RequestMessage Class Reference	179
7.59.1	Detailed Description	180
7.59.2	Member Function Documentation	180
7.59.2.1	addRequest	180
7.59.2.2	buffer	180
7.59.2.3	header	180
7.59.2.4	size	181
7.60	artdaq::detail::RequestPacket Struct Reference	181
7.60.1	Detailed Description	181
7.60.2	Constructor & Destructor Documentation	181
7.60.2.1	RequestPacket	181
7.60.3	Member Function Documentation	182
7.60.3.1	isValid	182
7.60.4	Member Data Documentation	182
7.60.4.1	header	182
7.61	artdaq::RequestSender Class Reference	182
7.61.1	Detailed Description	183
7.61.2	Constructor & Destructor Documentation	183
7.61.2.1	RequestSender	183
7.61.3	Member Function Documentation	183
7.61.3.1	AddRequest	183
7.61.3.2	GetRequestMethod	184
7.61.3.3	operator=	184
7.61.3.4	RemoveRequest	184
7.61.3.5	SendRequest	184
7.61.3.6	SendRoutingToken	184
7.61.3.7	SetRequestMethod	185
7.62	artdaq::resume_ Class Reference	185
7.62.1	Detailed Description	185
7.62.2	Constructor & Destructor Documentation	186
7.62.2.1	resume_	186
7.62.3	Member Data Documentation	186

7.62.3.1 defaultTimeout	186
7.62.3.2 defaultTimestamp	186
7.63 art::RootMPIOutput Class Reference	186
7.63.1 Detailed Description	187
7.63.2 Constructor & Destructor Documentation	187
7.63.2.1 RootMPIOutput	187
7.64 artdaq::RoundRobinPolicy Class Reference	187
7.64.1 Detailed Description	188
7.64.2 Constructor & Destructor Documentation	188
7.64.2.1 RoundRobinPolicy	188
7.64.3 Member Function Documentation	188
7.64.3.1 GetCurrentTable	188
7.65 artdaq::detail::RoutingAckPacket Struct Reference	188
7.65.1 Detailed Description	189
7.66 artdaq::RoutingMasterApp Class Reference	189
7.66.1 Detailed Description	190
7.66.2 Constructor & Destructor Documentation	190
7.66.2.1 RoutingMasterApp	190
7.66.3 Member Function Documentation	190
7.66.3.1 BootedEnter	190
7.66.3.2 do_initialize	190
7.66.3.3 do_pause	191
7.66.3.4 do_reinitialize	191
7.66.3.5 do_resume	191
7.66.3.6 do_shutdown	192
7.66.3.7 do_soft_initialize	192
7.66.3.8 do_start	192
7.66.3.9 do_stop	193
7.66.3.10 operator=	193
7.66.3.11 report	193
7.67 artdaq::RoutingMasterCore Class Reference	194
7.67.1 Detailed Description	195
7.67.2 Constructor & Destructor Documentation	195
7.67.2.1 RoutingMasterCore	195
7.67.2.2 ~RoutingMasterCore	195
7.67.3 Member Function Documentation	195
7.67.3.1 initialize	195

7.67.3.2 <code>operator=</code>	196
7.67.3.3 <code>pause</code>	196
7.67.3.4 <code>process_event_table</code>	196
7.67.3.5 <code>reinitialize</code>	196
7.67.3.6 <code>report</code>	197
7.67.3.7 <code>resume</code>	197
7.67.3.8 <code>send_event_table</code>	197
7.67.3.9 <code>shutdown</code>	197
7.67.3.10 <code>soft_initialize</code>	198
7.67.3.11 <code>start</code>	199
7.67.3.12 <code>stop</code>	199
7.68 <code>artdaq::RoutingMasterPolicy</code> Class Reference	199
7.68.1 Detailed Description	200
7.68.2 Constructor & Destructor Documentation	200
7.68.2.1 <code>RoutingMasterPolicy</code>	200
7.68.3 Member Function Documentation	201
7.68.3.1 <code>AddReceiverToken</code>	201
7.68.3.2 <code>GetCurrentTable</code>	201
7.68.3.3 <code>GetMaxNumberOfTokens</code>	201
7.68.3.4 <code>GetReceiverCount</code>	201
7.69 <code>RoutingMasterTest</code> Class Reference	201
7.69.1 Detailed Description	202
7.69.2 Constructor & Destructor Documentation	202
7.69.2.1 <code>RoutingMasterTest</code>	202
7.69.3 Member Function Documentation	203
7.69.3.1 <code>getPset</code>	203
7.70 <code>artdaq::detail::RoutingPacketEntry</code> Struct Reference	203
7.70.1 Detailed Description	203
7.70.2 Constructor & Destructor Documentation	204
7.70.2.1 <code>RoutingPacketEntry</code>	204
7.71 <code>artdaq::detail::RoutingPacketHeader</code> Struct Reference	205
7.71.1 Detailed Description	205
7.71.2 Constructor & Destructor Documentation	205
7.71.2.1 <code>RoutingPacketHeader</code>	205
7.72 <code>artdaq::detail::RoutingToken</code> Struct Reference	206
7.72.1 Detailed Description	206
7.73 <code>artdaq::RTIDDS</code> Class Reference	206

7.73.1	Detailed Description	207
7.73.2	Constructor & Destructor Documentation	207
7.73.2.1	RTIDDS	207
7.73.3	Member Function Documentation	207
7.73.3.1	copyFragmentToDDS_	207
7.73.3.2	moveFragmentToDDS_	208
7.74	artdaq::RTIDDSTransfer Class Reference	208
7.74.1	Detailed Description	209
7.74.2	Constructor & Destructor Documentation	209
7.74.2.1	RTIDDSTransfer	209
7.74.3	Member Function Documentation	209
7.74.3.1	copyFragment	209
7.74.3.2	moveFragment	209
7.74.3.3	receiveFragment	210
7.75	artdaq::SharedMemoryEventManager Class Reference	210
7.75.1	Detailed Description	212
7.75.2	Constructor & Destructor Documentation	212
7.75.2.1	SharedMemoryEventManager	212
7.75.3	Member Function Documentation	213
7.75.3.1	AddFragment	213
7.75.3.2	DoneWritingFragment	213
7.75.3.3	endOfData	213
7.75.3.4	endRun	213
7.75.3.5	endSubrun	214
7.75.3.6	GetArtEventCount	214
7.75.3.7	GetBroadcastKey	214
7.75.3.8	GetDroppedDataAddress	214
7.75.3.9	GetFragmentCount	214
7.75.3.10	GetInactiveEventCount	215
7.75.3.11	GetIncompleteEventCount	215
7.75.3.12	GetLockedBufferCount	215
7.75.3.13	GetPendingEventCount	215
7.75.3.14	ReconfigureArt	215
7.75.3.15	runID	216
7.75.3.16	setOverwrite	216
7.75.3.17	setRequestMethod	216
7.75.3.18	ShutdownArtProcesses	216

7.75.3.19 StartArtProcess	216
7.75.3.20 startRun	217
7.75.3.21 subrunID	217
7.75.3.22 WriteFragmentHeader	217
7.76 artdaq::detail::SharedMemoryReader< getDefaultTypes > Struct Template Reference	218
7.76.1 Detailed Description	219
7.76.2 Constructor & Destructor Documentation	219
7.76.2.1 SharedMemoryReader	219
7.76.2.2 SharedMemoryReader	219
7.76.3 Member Function Documentation	220
7.76.3.1 hasMoreData	220
7.76.3.2 operator=	220
7.76.3.3 readFile	220
7.76.3.4 readNext	220
7.77 artdaq::ShmemTransfer Class Reference	221
7.77.1 Detailed Description	222
7.77.2 Constructor & Destructor Documentation	222
7.77.2.1 ShmemTransfer	222
7.77.3 Member Function Documentation	222
7.77.3.1 copyFragment	222
7.77.3.2 moveFragment	222
7.77.3.3 receiveFragment	223
7.77.3.4 receiveFragmentData	223
7.77.3.5 receiveFragmentHeader	223
7.78 ShmRTTestFixture Struct Reference	224
7.78.1 Detailed Description	224
7.78.2 Member Function Documentation	225
7.78.2.1 getBroadcastKey	225
7.78.2.2 getKey	225
7.78.2.3 gf	225
7.78.2.4 helper	225
7.78.2.5 reader	225
7.78.2.6 source_helper	226
7.78.2.7 writer	226
7.79 artdaq::shutdown_ Class Reference	226
7.79.1 Detailed Description	227
7.79.2 Constructor & Destructor Documentation	227

7.79.2.1	shutdown_	227
7.79.3	Member Data Documentation	227
7.79.3.1	defaultTimeout	227
7.80	artdaq::soft_init_ Class Reference	227
7.80.1	Detailed Description	228
7.80.2	Constructor & Destructor Documentation	228
7.80.2.1	soft_init_	228
7.80.3	Member Data Documentation	228
7.80.3.1	defaultTimeout	228
7.80.3.2	defaultTimestamp	228
7.81	art::Source_generator< artdaq::detail::SharedMemoryReader<> > Struct Template Reference	228
7.81.1	Detailed Description	229
7.82	art::Source_generator< ArtdaqInput< artdaq::TransferWrapper > > Struct Template Reference	229
7.82.1	Detailed Description	229
7.83	art::Source_generator< ArtdaqInput< NetMonWrapper > > Struct Template Reference	229
7.83.1	Detailed Description	230
7.84	artdaq::start_ Class Reference	230
7.84.1	Detailed Description	230
7.84.2	Constructor & Destructor Documentation	230
7.84.2.1	start_	230
7.84.3	Member Data Documentation	231
7.84.3.1	defaultTimeout	231
7.84.3.2	defaultTimestamp	231
7.85	artdaq::StatisticsHelper Class Reference	231
7.85.1	Detailed Description	232
7.85.2	Member Function Documentation	232
7.85.2.1	addMonitoredQuantityName	232
7.85.2.2	addSample	232
7.85.2.3	createCollectors	232
7.85.2.4	operator=	233
7.85.2.5	readyToReport	233
7.85.2.6	statsRollingWindowHasMoved	233
7.86	artdaq::status_ Class Reference	234
7.86.1	Detailed Description	234
7.86.2	Constructor & Destructor Documentation	234
7.86.2.1	status_	234
7.87	artdaq::stop_ Class Reference	234

7.87.1	Detailed Description	235
7.87.2	Constructor & Destructor Documentation	235
7.87.2.1	stop_	235
7.87.3	Member Data Documentation	235
7.87.3.1	defaultTimeout	235
7.87.3.2	defaultTimestamp	235
7.88	artdaq::TCPSocketTransfer Class Reference	236
7.88.1	Detailed Description	236
7.88.2	Constructor & Destructor Documentation	236
7.88.2.1	TCPSocketTransfer	236
7.88.3	Member Function Documentation	237
7.88.3.1	copyFragment	237
7.88.3.2	moveFragment	237
7.88.3.3	receiveFragmentData	237
7.88.3.4	receiveFragmentHeader	238
7.89	anonymous_namespace{genToArt.cc}::ThrottledGenerator Class Reference	238
7.89.1	Detailed Description	239
7.89.2	Constructor & Destructor Documentation	239
7.89.2.1	ThrottledGenerator	239
7.89.3	Member Function Documentation	239
7.89.3.1	getNext	239
7.89.3.2	numFragIDs	239
7.89.3.3	start	239
7.89.3.4	stop	240
7.90	Timeout Class Reference	240
7.90.1	Detailed Description	241
7.90.2	Constructor & Destructor Documentation	241
7.90.2.1	Timeout	241
7.90.3	Member Function Documentation	241
7.90.3.1	add_periodic	241
7.90.3.2	add_periodic	242
7.90.3.3	add_periodic	243
7.90.3.4	add_relative	243
7.90.3.5	add_relative	243
7.90.3.6	cancel_timeout	243
7.90.3.7	copy_in_timeout	244
7.90.3.8	get_next_expired_timeout	244

7.90.3.9 <code>get_next_timeout_delay</code>	244
7.90.3.10 <code>get_next_timeout_msdl</code>	244
7.90.3.11 <code>is_consistent</code>	245
7.91 <code>Timeout::timeoutspec</code> Struct Reference	245
7.91.1 Detailed Description	245
7.92 <code>artdaq::TransferInterface</code> Class Reference	246
7.92.1 Detailed Description	247
7.92.2 Member Enumeration Documentation	247
7.92.2.1 <code>CopyStatus</code>	247
7.92.2.2 <code>Role</code>	247
7.92.3 Constructor & Destructor Documentation	248
7.92.3.1 <code>TransferInterface</code>	248
7.92.4 Member Function Documentation	248
7.92.4.1 <code>copyFragment</code>	248
7.92.4.2 <code>destination_rank</code>	248
7.92.4.3 <code>moveFragment</code>	248
7.92.4.4 <code>operator=</code>	249
7.92.4.5 <code>receiveFragment</code>	249
7.92.4.6 <code>receiveFragmentData</code>	249
7.92.4.7 <code>receiveFragmentHeader</code>	250
7.92.4.8 <code>role</code>	250
7.92.4.9 <code>source_rank</code>	250
7.92.4.10 <code>uniqueLabel</code>	250
7.93 <code>art::TransferOutput</code> Class Reference	251
7.93.1 Detailed Description	251
7.93.2 Constructor & Destructor Documentation	251
7.93.2.1 <code>TransferOutput</code>	251
7.94 <code>artdaq::TransferTest</code> Class Reference	252
7.94.1 Detailed Description	252
7.94.2 Constructor & Destructor Documentation	252
7.94.2.1 <code>TransferTest</code>	252
7.94.3 Member Function Documentation	253
7.94.3.1 <code>runTest</code>	253
7.95 <code>artdaq::TransferWrapper</code> Class Reference	253
7.95.1 Detailed Description	253
7.95.2 Constructor & Destructor Documentation	253
7.95.2.1 <code>TransferWrapper</code>	253

7.95.3 Member Function Documentation	254
7.95.3.1 receiveInitMessage	254
7.95.3.2 receiveMessage	254
7.96 artdaq::unregister_monitor_ Class Reference	254
7.96.1 Detailed Description	255
7.96.2 Constructor & Destructor Documentation	255
7.96.2.1 unregister_monitor_	255
7.97 artdaq::xmlrpc_commander Class Reference	255
7.97.1 Detailed Description	256
7.97.2 Constructor & Destructor Documentation	256
7.97.2.1 xmlrpc_commander	256
7.97.3 Member Function Documentation	256
7.97.3.1 send_register_monitor	256
7.97.3.2 send_unregister_monitor	257
8 File Documentation	259
8.1 artdaq/artdaq/DAQdata/TCP_listen_fd.hh File Reference	259
8.1.1 Detailed Description	259
8.1.2 Function Documentation	259
8.1.2.1 TCP_listen_fd	259
8.2 artdaq/artdaq/DAQdata/TCPConnect.hh File Reference	259
8.2.1 Detailed Description	260
8.2.2 Function Documentation	260
8.2.2.1 ResolveHost	260
8.2.2.2 ResolveHost	260
8.2.2.3 TCPConnect	260
8.3 artdaq/artdaq/DAQrate/quiet_mpi.hh File Reference	261
8.3.1 Detailed Description	261
Index	262

Chapter 1

Todo List

Member `artdaq::DataLoggerCore::DataLoggerCore (int rank, std::string name)`

Make the global queue size configurable

Member `artdaq::DispatcherCore::DispatcherCore (int rank, std::string name)`

Make the global queue size configurable

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

anonymous_namespace{genToArt.cc}	19
anonymous_namespace{rootOutputConfigurationTools.cc}	20
anonymous_namespace{shared_memory_reader_t.cc}	20
anonymous_namespace{TransferWrapper.cc}	21
art	
Namespace used for classes that interact directly with art	21
artdaq	
The artdaq namespace	23
artdaq::detail	
The <code>artdaq::detail</code> namespace contains internal implementation details for some classes	33

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

artdaq::art_config_file	37
art::ArtdaqInput< U >	38
artdaq::BoardReaderCore	51
artdaq::Commandable	64
artdaq::BoardReaderApp	46
artdaq::DataLoggerApp	94
artdaq::DispatcherApp	112
artdaq::EventBuilderApp	122
artdaq::RoutingMasterApp	189
artdaq::CommanderInterface	87
artdaq::xmlrpc_commander	255
artdaq::DataReceiverCore	101
artdaq::DataLoggerCore	99
artdaq::DispatcherCore	117
artdaq::EventBuilderCore	126
artdaq::DataReceiverManager	107
artdaq::DataSenderManager	109
DDSDataReaderListener	
artdaq::RTIDDS::OctetsListener	169
art::detail::DummyProductCache	121
EDAnalyzer	
artdaq::EventDump	128
artdaq::FragmentSniffer	135
artdaq::PrintBuildInfo	171
EDFilter	
artdaq::RandomDelayFilter	173
EDProducer	
artdaq::BuildInfo< instanceName, Pkgs >	57
artdaq::detail::FragCounter	130
FragmentGenerator	
artdaq::CommandableFragmentGenerator	73
artdaq::CompositeDriver	93
artdaqtest::CommandableFragmentGeneratorTest	85

artdaq::GenericFragmentSimulator	139
artdaq::FragmentReceiverManager	132
artdaq::FragmentStoreElement	137
artdaq::GetPackageBuildInfo	142
artdaq::Globals	142
LockFile	145
MessHead	147
method	
artdaq::cmd_	60
artdaq::init_	143
artdaq::legal_commands_	145
artdaq::pause_	170
artdaq::register_monitor_	175
artdaq::reinit_	176
artdaq::report_	177
artdaq::resume_	185
artdaq::shutdown_	226
artdaq::soft_init_	227
artdaq::start_	230
artdaq::status_	234
artdaq::stop_	234
artdaq::unregister_monitor_	254
MPIProg	148
Builder	56
RoutingMasterTest	201
artdaq::MPISentry	149
MPRGlobalTestFixture	153
artdaq::NetMonHeader	159
NetMonTransportServiceInterface	162
NetMonTransportService	160
art::NetMonWrapper	164
OutputModule	
art::BinaryFileOutput	44
art::BinaryMPIOutput	45
art::RootMPIOutput	186
art::TransferOutput	251
daqrate.Re	175
artdaq::detail::RequestHeader	178
artdaq::detail::RequestMessage	179
artdaq::detail::RequestPacket	181
artdaq::RequestSender	182
artdaq::detail::RoutingAckPacket	188
artdaq::RoutingMasterCore	194
artdaq::RoutingMasterPolicy	199
artdaq::CapacityTestPolicy	59
artdaq::NoOpPolicy	165
artdaq::RoundRobinPolicy	187
artdaq::detail::RoutingPacketEntry	203
artdaq::detail::RoutingPacketHeader	205
artdaq::detail::RoutingToken	206
artdaq::RTIDDS	206
SharedMemoryManager	
artdaq::SharedMemoryEventManager	210

artdaq::detail::SharedMemoryReader< getDefaultTypes >	218
ShmRTestFixture	224
art::Source_generator< artdaq::detail::SharedMemoryReader<> >	228
art::Source_generator< ArtdaqInput< artdaq::TransferWrapper > >	229
art::Source_generator< ArtdaqInput< NetMonWrapper > >	229
artdaq::StatisticsHelper	231
anonymous_namespace{genToArt.cc}::ThrottledGenerator	238
Timeout	240
Timeout::timeoutspec	245
artdaq::TransferInterface	246
artdaq::AutodetectTransfer	40
artdaq::MPITransfer	151
artdaq::MulticastTransfer	155
artdaq::NullTransfer	166
artdaq::RTIDDSTransfer	208
artdaq::ShmemTransfer	221
artdaq::TCPSocketTransfer	236
artdaq::TransferTest	252
artdaq::TransferWrapper	253

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

artdaq::art_config_file	
Art_config_file wraps a temporary file used to configure art	37
art::ArtdaqInput< U >	
This template class provides a unified interface for reading data into art	38
artdaq::AutodetectTransfer	
The AutodetectTransfer TransferInterface plugin sets up a Shmem_transfer plugin or TCPSocket_transfer plugin depending if the source and destination are on the same host, to maximize throughput	40
art::BinaryFileOutput	
The BinaryFileOutput module streams art Events to a binary file, bypassing ROOT	44
art::BinaryMPIOutput	
An art::OutputModule which sends Fragments using DataSenderManager. This module produces output identical to that of a BoardReader, for use in systems which have multiple layers of EventBuilders	45
artdaq::BoardReaderApp	
BoardReaderApp is an artdaq::Commandable derived class which controls the BoardReaderCore state machine	46
artdaq::BoardReaderCore	
BoardReaderCore implements the state machine for the BoardReader artdaq application. It contains a CommandableFragmentGenerator , which generates Fragments which are then sent to a Data-SenderManager by BoardReaderCore	51
Builder	
Runs the builder test	56
artdaq::BuildInfo< instanceName, Pkgs >	
BuildInfo is an art::EDProducer which saves information about package builds to the data file	57
artdaq::CapacityTestPolicy	
A RoutingMasterPolicy which tries to fully load the first receiver, then the second, and so on	59
artdaq::cmd_	
The "cmd_" class serves as the base class for all artdaq's XML-RPC commands	60
artdaq::Commandable	
Commandable is the base class for all artdaq components which implement the artdaq state machine	64
artdaq::CommandableFragmentGenerator	
CommandableFragmentGenerator is a FragmentGenerator-derived abstract class that defines the interface for a FragmentGenerator designed as a state machine with start, stop, etc., transition commands	73

artdaqtest::CommandableFragmentGeneratorTest	CommandableFragmentGenerator derived class for testing	85
artdaq::CommanderInterface	This interface defines the functions used to transfer data between artdaq applications	87
artdaq::CompositeDriver	CompositeDriver handles a set of lower-level generators	93
artdaq::DataLoggerApp	DataLoggerApp is an artdaq::Commandable derived class which controls the DataLoggerCore	94
artdaq::DataLoggerCore	DataLoggerCore implements the state machine for the DataLogger artdaq application. DataLoggerCore processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher	99
artdaq::DataReceiverCore	DataReceiverCore implements the state machine for the DataReceiver artdaq application. DataReceiverCore receives Fragment objects from the DataReceiverManager , and sends them to the EventStore	101
artdaq::DataReceiverManager	Receives Fragment objects from one or more DataSenderManager instances using TransferInterface plugins DataReceiverMaanger runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others	107
artdaq::DataSenderManager	Sends Fragment objects using TransferInterface plugins. Uses Routing Tables if configiured, otherwise will Round-Robin Fragments to the destinations	109
artdaq::DispatcherApp	DispatcherApp is an artdaq::Commandable derived class which controls the DispatcherCore	112
artdaq::DispatcherCore	DispatcherCore implements the state machine for the Dispatcher artdaq application. DispatcherCore processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher	117
art::detail::DummyProductCache	A lightweight product cache for when the full thing is not appropriate (TBB and ROOT don't fully get along)	121
artdaq::EventBuilderApp	EventBuilderApp is an artdaq::Commandable derived class which controls the EventBuilderCore	122
artdaq::EventBuilderCore	EventBuilderCore implements the state machine for the EventBuilder artdaq application. EventBuilderCore receives Fragment objects from the DataReceiverManager , and sends them to the EventStore	126
artdaq::EventDump	Write Event information to the console	128
artdaq::detail::FragCounter	Keep track of the count of Fragments received from a set of sources	130
artdaq::FragmentReceiverManager	Receives Fragment objects from one or more DataSenderManager instances using TransferInterface plugins DataReceiverMaanger runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others	132
artdaq::FragmentSniffer	This art::EDAnalyzer plugin tries to get Fragments from each event, asserting that the correct number of Fragments were present	135
artdaq::FragmentStoreElement	This class contains tracking information for all Fragment objects which have been received from a specific source	137
artdaq::GenericFragmentSimulator	GenericFragmentSimulator creates simulated Generic events, with data distributed according to a "histogram" provided in the configuration data	139

artdaq::GetPackageBuildInfo	Wrapper around the <code>artdaq::GetPackageBuildInfo::getPackageBuildInfo</code> function	142
artdaq::Globals	The <code>artdaq::Globals</code> class contains several variables which are useful across the entire artdaq system	142
artdaq::init_	143
artdaq::legal_commands_	Legal_commands_ Command class	145
LockFile	Create a "lock file", removing it upon class destruction	145
MessHead	This header is sent by the <code>TCP Socket_transfer</code> to allow for more efficient writev calls	147
MPIProg	A wrapper for a MPI program. Similar to <code>MPIISentry</code>	148
artdaq::MPIISentry	Initializes and finalizes the MPI context that the artdaq applications run in	149
artdaq::MPITransfer	<code>MPITransfer</code> is a <code>TransferInterface</code> implementation plugin that transfers data using MPI	151
MPRGlobalTestFixture	MasterProductRegistry Test Fixture	153
artdaq::MulticastTransfer	<code>MulticastTransfer</code> is a <code>TransferInterface</code> implementation plugin that transfers data using Multicast	155
artdaq::NetMonHeader	Header with length information for <code>NetMonTransport</code> messages	159
NetMonTransportService	<code>NetMonTransportService</code> extends <code>NetMonTransportServiceInterface</code> . It sends events using Data-SenderManager and receives events from the GlobalQueue	160
NetMonTransportServiceInterface	Interface for <code>NetMonTransportService</code> . This interface is declared to art as part of the required registration of an art Service	162
art::NetMonWrapper	This class wraps <code>NetMonTransportService</code> so that it can act as an <code>ArtdaqInput</code> template class	164
artdaq::NoOpPolicy	A <code>RoutingMasterPolicy</code> which simply assigns Sequence IDs to tokens in the order they were received	165
artdaq::NullTransfer	<code>NullTransfer</code> does not send or receive data, but acts as if it did	166
artdaq::RTIDDS::OctetsListener	A class that reads data from DDS	169
artdaq::pause_	170
artdaq::PrintBuildInfo	An art::EDAnalyzer which prints any <code>artdaq::BuildInfo</code> objects stored in the run	171
artdaq::RandomDelayFilter	A filter which delays for a random amount of time, then drops a random fraction of events. Used to simulate the delays and efficiency of real filters	173
daqrate.Re	General Regular Expression class that allows for: <code>xx = Re(re)</code>	175
artdaq::register_monitor_	Register_monitor_ Command class	175
artdaq::reinit_	176
artdaq::report_	Report_ Command class	177
artdaq::detail::RequestHeader	Header of a <code>RequestMessage</code> . Contains magic bytes for validation and a count of expected Request-Packets	178

artdaq::detail::RequestMessage	A RequestMessage consists of a RequestHeader and zero or more RequestPackets . They will usually be sent in two calls to send()	179
artdaq::detail::RequestPacket	The RequestPacket contains information about a single data request	181
artdaq::RequestSender	The RequestSender contains methods used to send data requests and Routing tokens	182
artdaq::resume_		185
art::RootMPIOutput	An art::OutputModule which sends events using DataSenderManager . This module is designed for transporting Fragment-wrapped art::Events after they have been read into art , for example between the EventBuilder and the Aggregator	186
artdaq::RoundRobinPolicy	A RoutingMasterPolicy which evenly distributes Sequence IDs to all receivers. If an uneven number of tokens have been received, extra tokens are stored for the next table update	187
artdaq::detail::RoutingAckPacket	A RoutingAckPacket contains the rank of the table receiver, plus the first and last sequence IDs in the Routing Table (for verification)	188
artdaq::RoutingMasterApp	RoutingMasterApp is an artdaq::Commandable derived class which controls the RoutingMasterCore state machine	189
artdaq::RoutingMasterCore	RoutingMasterCore implements the state machine for the RoutingMaster artdaq application. RoutingMasterCore collects tokens from receivers, and at regular intervals uses these tokens to build Routing Tables that are sent to the senders	194
artdaq::RoutingMasterPolicy	The interface through which RoutingMasterCore obtains Routing Tables using received Routing Tokens	199
RoutingMasterTest	Runs the routing_master test	201
artdaq::detail::RoutingPacketEntry	A row of the Routing Table	203
artdaq::detail::RoutingPacketHeader	The header of the Routing Table, containing the magic bytes and the number of entries	205
artdaq::detail::RoutingToken	The RoutingToken contains the magic bytes, the rank of the token sender, and the number of slots free. This is a TCP message, so additional verification is not necessary	206
artdaq::RTIDDS	DDS Transport Implementation	206
artdaq::RTIDDSTransfer	RTIDDSTransfer is a TransferInterface implementation plugin that transfers data using RTI DDS	208
artdaq::SharedMemoryEventManager	The SharedMemoryEventManager is a SharedMemoryManger which tracks events as they are built	210
artdaq::detail::SharedMemoryReader< getDefaultTypes >	The SharedMemoryReader is a class which implements the methods needed by art::Source	218
artdaq::ShmemTransfer	A TransferInterface implementation plugin that transfers data using Shared Memory	221
ShmRTestFixture	SharedMemoryReader Test Fixture	224
artdaq::shutdown_	Shutdown_ Command class	226
artdaq::soft_init_		227

art::Source_generator< artdaq::detail::SharedMemoryReader<> >	
Specialize an art source trait to tell art that we don't care about source.fileNames and don't want the files services to be used	228
art::Source_generator< ArtdaqInput< artdaq::TransferWrapper > >	
Trait definition (must precede source typedef)	229
art::Source_generator< ArtdaqInput< NetMonWrapper > >	
Trait definition (must precede source typedef)	229
artdaq::start_	
Command class representing a start transition	230
artdaq::StatisticsHelper	
This class manages MonitoredQuantity instances for the *Core classes	231
artdaq::status_	
Status_ Command class	234
artdaq::stop_	234
artdaq::TCPSocketTransfer	
TransferInterface implementation plugin that sends data using TCP sockets	236
anonymous_namespace{genToArt.cc}::ThrottledGenerator	
ThrottledGenerator: ensure that we only get one fragment per type at a time from the generator	238
Timeout	
Performs registered actions at specified intervals	240
Timeout::timeoutspec	
Specification for a Timeout function	245
artdaq::TransferInterface	
This interface defines the functions used to transfer data between artdaq applications	246
art::TransferOutput	
An art::OutputModule which sends events using DataSenderManager. This module is designed for transporting Fragment-wrapped art::Events after they have been read into art, for example between the EventBuilder and the Aggregator	251
artdaq::TransferTest	
Test a set of TransferInterface plugins	252
artdaq::TransferWrapper	
TransferWrapper wraps a TransferInterface so that it can be used in the ArtdaqInput class to make an art::Source	253
artdaq::unregister_monitor_	
Unregister_monitor_ Command class	254
artdaq::xmlrpc_commander	
Serves as the XMLRPC server run in each artdaq application	255

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

artdaq/artdaq/Application/ BoardReaderApp.cc	??
artdaq/artdaq/Application/ BoardReaderApp.hh	??
artdaq/artdaq/Application/ BoardReaderCore.cc	??
artdaq/artdaq/Application/ BoardReaderCore.hh	??
artdaq/artdaq/Application/ Commandable.cc	??
artdaq/artdaq/Application/ Commandable.hh	??
artdaq/artdaq/Application/ CommandableFragmentGenerator.cc	??
artdaq/artdaq/Application/ CommandableFragmentGenerator.hh	??
artdaq/artdaq/Application/ CompositeDriver.hh	??
artdaq/artdaq/Application/ CompositeDriver_generator.cc	??
artdaq/artdaq/Application/ DataLoggerApp.cc	??
artdaq/artdaq/Application/ DataLoggerApp.hh	??
artdaq/artdaq/Application/ DataLoggerCore.cc	??
artdaq/artdaq/Application/ DataLoggerCore.hh	??
artdaq/artdaq/Application/ DataReceiverCore.cc	??
artdaq/artdaq/Application/ DataReceiverCore.hh	??
artdaq/artdaq/Application/ DispatcherApp.cc	??
artdaq/artdaq/Application/ DispatcherApp.hh	??
artdaq/artdaq/Application/ DispatcherCore.cc	??
artdaq/artdaq/Application/ DispatcherCore.hh	??
artdaq/artdaq/Application/ EventBuilderApp.cc	??
artdaq/artdaq/Application/ EventBuilderApp.hh	??
artdaq/artdaq/Application/ EventBuilderCore.cc	??
artdaq/artdaq/Application/ EventBuilderCore.hh	??
artdaq/artdaq/Application/ GeneratorMacros.hh	??
artdaq/artdaq/Application/ LoadParameterSet.hh	??
artdaq/artdaq/Application/ makeCommandableFragmentGenerator.cc	??
artdaq/artdaq/Application/ makeCommandableFragmentGenerator.hh	??
artdaq/artdaq/Application/ RoutingMasterApp.cc	??
artdaq/artdaq/Application/ RoutingMasterApp.hh	??
artdaq/artdaq/Application/ RoutingMasterCore.cc	??
artdaq/artdaq/Application/ RoutingMasterCore.hh	??
artdaq/artdaq/Application/ StatisticsHelper.cc	??
artdaq/artdaq/Application/ StatisticsHelper.hh	??

artdaq/artdaq/Application/ TaskType.hh	??
artdaq/artdaq/Application/MPI2/ BoardReaderMain.cc	??
artdaq/artdaq/Application/MPI2/ DataLoggerMain.cc	??
artdaq/artdaq/Application/MPI2/ DispatcherMain.cc	??
artdaq/artdaq/Application/MPI2/ EventBuilderMain.cc	??
artdaq/artdaq/Application/MPI2/ MPIISentry.cc	??
artdaq/artdaq/Application/MPI2/ MPIISentry.hh	??
artdaq/artdaq/Application/MPI2/ RoutingMasterMain.cc	??
artdaq/artdaq/Application/Routing/ CapacityTest_policy.cc	??
artdaq/artdaq/Application/Routing/ makeRoutingMasterPolicy.cc	??
artdaq/artdaq/Application/Routing/ makeRoutingMasterPolicy.hh	??
artdaq/artdaq/Application/Routing/ NoOp_policy.cc	??
artdaq/artdaq/Application/Routing/ PolicyMacros.hh	??
artdaq/artdaq/Application/Routing/ RoundRobin_policy.cc	??
artdaq/artdaq/Application/Routing/ RoutingMasterPolicy.cc	??
artdaq/artdaq/Application/Routing/ RoutingMasterPolicy.hh	??
artdaq/artdaq/ArtModules/ ArtdaqBuildInfo_module.cc	??
artdaq/artdaq/ArtModules/ ArtdaqInput.hh	??
artdaq/artdaq/ArtModules/ BinaryFileOutput_module.cc	??
artdaq/artdaq/ArtModules/ BinaryMPIOutput_module.cc	??
artdaq/artdaq/ArtModules/ BuildInfo_module.hh	??
artdaq/artdaq/ArtModules/ classes.h	??
artdaq/artdaq/ArtModules/ EventDump_module.cc	??
artdaq/artdaq/ArtModules/ InputUtilities.hh	??
artdaq/artdaq/ArtModules/ NetMonInput_source.cc	??
artdaq/artdaq/ArtModules/ NetMonTransportService.h	??
artdaq/artdaq/ArtModules/ NetMonTransportService_service.cc	??
artdaq/artdaq/ArtModules/ NetMonTransportServiceInterface.h	??
artdaq/artdaq/ArtModules/ NetMonWrapper.cc	??
artdaq/artdaq/ArtModules/ NetMonWrapper.hh	??
artdaq/artdaq/ArtModules/ PrintBuildInfo_module.cc	??
artdaq/artdaq/ArtModules/ RandomDelayFilter_module.cc	??
artdaq/artdaq/ArtModules/ RawInput_source.cc	??
artdaq/artdaq/ArtModules/ RootMPIOutput_module.cc	??
artdaq/artdaq/ArtModules/ TransferInput_source.cc	??
artdaq/artdaq/ArtModules/ TransferOutput_module.cc	??
artdaq/artdaq/ArtModules/detail/ DummyProductCache.cc	??
artdaq/artdaq/ArtModules/detail/ DummyProductCache.h	??
artdaq/artdaq/ArtModules/detail/ LogFileAction.cc	??
artdaq/artdaq/ArtModules/detail/ LogFileAction.h	??
artdaq/artdaq/ArtModules/detail/ ParentageMap.hh	??
artdaq/artdaq/ArtModules/detail/ rootOutputConfigurationTools.cc	??
artdaq/artdaq/ArtModules/detail/ rootOutputConfigurationTools.h	??
artdaq/artdaq/ArtModules/detail/ SharedMemoryReader.hh	??
artdaq/artdaq/ArtModules/detail/ TransferWrapper.cc	??
artdaq/artdaq/ArtModules/detail/ TransferWrapper.hh	??
artdaq/artdaq/BuildInfo/ GetPackageBuildInfo.hh	??
artdaq/artdaq/DAQdata/ GenericFragmentSimulator.hh	??
artdaq/artdaq/DAQdata/ GenericFragmentSimulator_generator.cc	??
artdaq/artdaq/DAQdata/ Globals.cc	??
artdaq/artdaq/DAQdata/ Globals.hh	??
artdaq/artdaq/DAQdata/ NetMonHeader.hh	??
artdaq/artdaq/DAQdata/ TCP_listen_fd.cc	??
artdaq/artdaq/DAQdata/ TCP_listen_fd.hh	259

artdaq/artdaq/DAQdata/ TCPConnect.cc	??
artdaq/artdaq/DAQdata/ TCPConnect.hh	259
artdaq/artdaq/DAQrate/ DataReceiverManager.cc	??
artdaq/artdaq/DAQrate/ DataReceiverManager.hh	??
artdaq/artdaq/DAQrate/ DataSenderManager.cc	??
artdaq/artdaq/DAQrate/ DataSenderManager.hh	??
artdaq/artdaq/DAQrate/ infoFilename.cc	??
artdaq/artdaq/DAQrate/ infoFilename.hh	??
artdaq/artdaq/DAQrate/ quiet_mpi.hh	261
artdaq/artdaq/DAQrate/ RequestSender.cc	??
artdaq/artdaq/DAQrate/ RequestSender.hh	??
artdaq/artdaq/DAQrate/ SharedMemoryEventManager.cc	??
artdaq/artdaq/DAQrate/ SharedMemoryEventManager.hh	??
artdaq/artdaq/DAQrate/detail/ FragCounter.hh	??
artdaq/artdaq/DAQrate/detail/ RequestMessage.cc	??
artdaq/artdaq/DAQrate/detail/ RoutingPacket.cc	??
artdaq/artdaq/ExternalComms/ CommanderInterface.cc	??
artdaq/artdaq/ExternalComms/ CommanderInterface.hh	??
artdaq/artdaq/ExternalComms/ MakeCommanderPlugin.cc	??
artdaq/artdaq/ExternalComms/ MakeCommanderPlugin.hh	??
artdaq/artdaq/ExternalComms/ xmlrpc_commander.cc	??
artdaq/artdaq/ExternalComms/ xmlrpc_commander.hh	??
artdaq/artdaq/RTIDDS/ RTIDDS.cc	??
artdaq/artdaq/RTIDDS/ RTIDDS.hh	??
artdaq/artdaq/TransferPlugins/ Autodetect_transfer.cc	??
artdaq/artdaq/TransferPlugins/ MakeTransferPlugin.cc	??
artdaq/artdaq/TransferPlugins/ MakeTransferPlugin.hh	??
artdaq/artdaq/TransferPlugins/ MPI_transfer.cc	??
artdaq/artdaq/TransferPlugins/ MPITransfer.hh	??
artdaq/artdaq/TransferPlugins/ Multicast_transfer.cc	??
artdaq/artdaq/TransferPlugins/ Null_transfer.cc	??
artdaq/artdaq/TransferPlugins/ RTIDDS_transfer.cc	??
artdaq/artdaq/TransferPlugins/ Shmem_transfer.cc	??
artdaq/artdaq/TransferPlugins/ ShmemTransfer.hh	??
artdaq/artdaq/TransferPlugins/ TCPSocket_transfer.cc	??
artdaq/artdaq/TransferPlugins/ TCPSocketTransfer.hh	??
artdaq/artdaq/TransferPlugins/ TransferInterface.cc	??
artdaq/artdaq/TransferPlugins/ TransferInterface.hh	??
artdaq/artdaq/TransferPlugins/detail/ SR.Sockets.hh	??
artdaq/artdaq/TransferPlugins/detail/ Timeout.cc	??
artdaq/artdaq/TransferPlugins/detail/ Timeout.hh	??
artdaq/doc/sample_MPMD_runs/ cpiA.cc	??
artdaq/doc/sample_MPMD_runs/ cpiB.cc	??
artdaq/proto/ builder.cc	??
artdaq/proto/ daqrate.py	??
artdaq/proto/ datalogger.cc	??
artdaq/proto/ dispatcher.cc	??
artdaq/proto/ driver.cc	??
artdaq/proto/ fhi1l_test.cc	??
artdaq/proto/ FragmentReceiverManager.cc	??
artdaq/proto/ FragmentReceiverManager.hh	??
artdaq/proto/ MPIProg.hh	??
artdaq/proto/ PrintSharedMemory.cc	??
artdaq/proto/ routing_master.cc	??

artdaq/proto/tracemf.cc	??
artdaq/proto/transfer_driver.cc	??
artdaq/proto/transfer_driver_mpi.cc	??
artdaq/proto/transfer_plugin_receiver.cc	??
artdaq/proto/transfer_plugin_sender.cc	??
artdaq/proto/TransferTest.cc	??
artdaq/proto/TransferTest.hh	??
artdaq/test/Application/CommandableFragmentGenerator_t.cc	??
artdaq/test/Application/Routing/CapacityTest_policy_t.cc	??
artdaq/test/Application/Routing/NoOp_policy_t.cc	??
artdaq/test/Application/Routing/RoundRobin_policy_t.cc	??
artdaq/test/ArtModules/daq_flow_t.cc	??
artdaq/test/ArtModules/FragmentSniffer_module.cc	??
artdaq/test/ArtModules/reconfigure_t.cc	??
artdaq/test/ArtModules/shared_memory_reader_t.cc	??
artdaq/test/DAQdata/GenericFragmentSimulator_t.cc	??
artdaq/test/DAQrate/DataReceiverManager_t.cc	??
artdaq/test/DAQrate/FragCounter_t.cc	??
artdaq/test/DAQrate/RequestSender_t.cc	??
artdaq/test/DAQrate/SharedMemoryEventManager_t.cc	??
artdaq/tools/genToArt.cc	??
artdaq/tools/StateResponder.cc	??

Chapter 6

Namespace Documentation

6.1 anonymous_namespace{genToArt.cc} Namespace Reference

Classes

- class [ThrottledGenerator](#)

ThrottledGenerator: ensure that we only get one fragment per type from the generator.

Functions

- int [process_cmd_line](#) (int argc, char **argv, bpo::variables_map &vm)
Process the command line.
- int [process_data](#) (fhicl::ParameterSet const &pset)
Run the test, instantiating configured generators and an EventStore.

6.1.1 Function Documentation

6.1.1.1 int anonymous_namespace{genToArt.cc}::process_cmd_line (int argc, char ** argv, bpo::variables_map & vm)

Process the command line.

Parameters

	<i>argc</i>	Number of arguments
	<i>argv</i>	Array of arguments as strings
<i>out</i>	<i>vm</i>	Output boost::program_options::variables_map

Returns

0 if success, -1 if exception, 1 if help was requested, and 2 if missing required arguments

Definition at line 47 of file genToArt.cc.

6.1.1.2 int anonymous_namespace{genToArt.cc}::process_data (fhicl::ParameterSet const & pset)

Run the test, instantiating configured generators and an EventStore.

Parameters

<i>pset</i>	ParameterSet used to configure genToArt
-------------	---

Returns

Art return code, of 15 if EventStore::endOfData fails

```
* genToArt accepts the following Parameters:
* "reset_sequenceID" (Default: true): Set the sequence IDs on generated Fragment objects to the expected value
* "genToArt" (REQUIRED): FHiCL table containing genToArt parameters
* "fragment_receivers" (REQUIRED): List of FHiCL tables configuring the Fragment receivers
* Each table should contain parameter "generator", the FragmentGenerator plugin to load, and any other parameters.
* "event_builder" (Default: {}): ParameterSet for EventStore. See documentation for configuration parameters.
* "run_number" (REQUIRED): Run number to use
* "events_to_generate" (Default: -1): Number of events to generate
*
*
```

Definition at line 222 of file genToArt.cc.

6.2 anonymous_namespace{rootOutputConfigurationTools.cc} Namespace Reference

Functions

- bool [maxCriterionSpecified](#) (ClosingCriteria const &cc)

Determine if a file-closing criteria has been provided.

6.2.1 Function Documentation

6.2.1.1 bool anonymous_namespace{rootOutputConfigurationTools.cc}::maxCriterionSpecified (ClosingCriteria const & cc)

Determine if a file-closing criteria has been provided.

Parameters

<i>cc</i>	ClosingCriteria to examine
-----------	----------------------------

Returns

Whether any criteria have been specified in the given ClosingCriteria instance

Definition at line 21 of file rootOutputConfigurationTools.cc.

6.3 anonymous_namespace{shared_memory_reader_t.cc} Namespace Reference

Functions

- void [basic_test](#) (artdaq::detail::SharedMemoryReader<> &reader, artdaq::SharedMemoryEventManager &writer, std::unique_ptr< art::RunPrincipal > &&run, std::unique_ptr< art::SubRunPrincipal > &&subrun, art::EventID const &eventid)

Run a basic checkout of the SharedMemoryReader.

6.3.1 Function Documentation

6.3.1.1 void anonymous_namespace{shared_memory_reader_t.cc}::basic_test (`artdaq::detail::SharedMemoryReader<>`
`& reader, artdaq::SharedMemoryEventManager & writer, std::unique_ptr<art::RunPrincipal> && run,`
`std::unique_ptr<art::SubRunPrincipal> && subrun, art::EventID const & eventid)`

Run a basic checkout of the SharedMemoryReader.

Parameters

<code>reader</code>	SharedMemoryReader instance
<code>writer</code>	SharedMemoryWriter instance
<code>run</code>	Run principal pointer
<code>subrun</code>	Subrun principal pointer
<code>eventid</code>	ID of event

Definition at line 328 of file shared_memory_reader_t.cc.

6.4 anonymous_namespace{TransferWrapper.cc} Namespace Reference

Variables

- volatile std::sig_atomic_t [gSignalStatus](#) = 0

Stores singal from signal handler.

6.5 art Namespace Reference

Namespace used for classes that interact directly with art.

Classes

- class [ArtdaqInput](#)

This template class provides a unified interface for reading data into art.

- class [BinaryFileOutput](#)

The BinaryFileOutput module streams art Events to a binary file, bypassing ROOT.

- class [BinaryMPIOutput](#)

An art::OutputModule which sends Fragments using DataSenderManager. This module produces output identical to that of a BoardReader, for use in systems which have multiple layers of EventBuilders.

- struct [Source_generator< ArtdaqInput< NetMonWrapper > >](#)

Trait definition (must precede source typedef).

- class [NetMonWrapper](#)

This class wraps NetMonTransportService so that it can act as an ArtdaqInput template class.

- struct [Source_generator< artdaq::detail::SharedMemoryReader > >](#)

Specialize an art source trait to tell art that we don't care about source.fileNames and don't want the files services to be used.

- class [RootMPIOutput](#)

An art::OutputModule which sends events using DataSenderManager. This module is designed for transporting Fragment-wrapped art::Events after they have been read into art, for example between the EventBuilder and the Aggregator.

- struct [Source_generator< ArtdaqInput< artdaq::TransferWrapper > >](#)

Trait definition (must precede source typeid).

- class [TransferOutput](#)

An art::OutputModule which sends events using DataSenderManager. This module is designed for transporting Fragment-wrapped art::Events after they have been read into art, for example between the EventBuilder and the Aggregator.

Typedefs

- `typedef std::map< ParentageID
const, Parentage > ParentageMap`

An art::ParentageMap, defined using a std::map.

- `typedef art::Source
< ArtdaqInput< NetMonWrapper > > NetMonInput`

NetMonInput is an art::Source using an ArtdaqInput-wrapped NetMonWrapper.

- `typedef art::Source
< ArtdaqInput
< artdaq::TransferWrapper > > TransferInput`

TransferInput is an art::Source using the artdaq::TransferWrapper class as the data source.

Functions

- `template<typename T >
T * ReadObjectAny (const std::unique_ptr< TBufferFile > &infile, const std::string &className, const std::string &callerName)`

ReadObjectAny reads data from a TBufferFile and casts it to the given type.

- `template<typename T >`

`void printProcessHistoryID (const std::string &label, const T &object)`

Print the processHistoryID from the object.

- `template<typename T >`

`void printProcessMap (const T &mappable, const std::string description)`

Print data from a map-like class.

6.5.1 Detailed Description

Namespace used for classes that interact directly with art.

6.5.2 Function Documentation

6.5.2.1 `template<typename T > void art::printProcessHistoryID (const std::string & label, const T & object)`

Print the processHistoryID from the object.

Template Parameters

<i>T</i>	Type of the object
----------	--------------------

Parameters

<i>label</i>	Label for the object
<i>object</i>	Object to print processHistoryID from

Definition at line 86 of file InputUtilities.hh.

6.5.2.2 template<typename T > void art::printProcessMap (const T & *mappable*, const std::string *description*)

Print data from a map-like class.

Template Parameters

<i>T</i>	Type of the class
----------	-------------------

Parameters

<i>mappable</i>	Map-like class to print
<i>description</i>	Description of the map-like class

Definition at line 117 of file InputUtilities.hh.

6.5.2.3 template<typename T > T* art::ReadObjectAny (const std::unique_ptr< TBufferFile > & *infile*, const std::string & *className*, const std::string & *callerName*)

ReadObjectAny reads data from a TBufferFile and casts it to the given type.

Template Parameters

<i>T</i>	The type of the data being read
----------	---------------------------------

Parameters

<i>infile</i>	A pointer to the TBufferFile being read
<i>className</i>	Name of the class to retrieve (must be in ROOT dictionary)
<i>callerName</i>	Name of the calling class, for logging purposes

Returns

Pointer to object of type T

Definition at line 52 of file InputUtilities.hh.

6.6 artdaq Namespace Reference

The artdaq namespace.

Namespaces

- [detail](#)

The artdaq::detail namespace contains internal implementation details for some classes.

Classes

- class [BoardReaderApp](#)
BoardReaderApp is an `artdaq::Commandable` derived class which controls the `BoardReaderCore` state machine.
- class [BoardReaderCore](#)
BoardReaderCore implements the state machine for the BoardReader artdaq application. It contains a `CommandableFragmentGenerator`, which generates Fragments which are then sent to a `DataSenderManager` by `BoardReaderCore`.
- class [Commandable](#)
`Commandable` is the base class for all artdaq components which implement the artdaq state machine.
- class [CommandableFragmentGenerator](#)
`CommandableFragmentGenerator` is a FragmentGenerator-derived abstract class that defines the interface for a FragmentGenerator designed as a state machine with start, stop, etc., transition commands.
- class [CompositeDriver](#)
`CompositeDriver` handles a set of lower-level generators.
- class [DataLoggerApp](#)
`DataLoggerApp` is an `artdaq::Commandable` derived class which controls the `DataLoggerCore`.
- class [DataLoggerCore](#)
`DataLoggerCore` implements the state machine for the DataLogger artdaq application. `DataLoggerCore` processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.
- class [DataReceiverCore](#)
`DataReceiverCore` implements the state machine for the DataReceiver artdaq application. `DataReceiverCore` receives Fragment objects from the `DataReceiverManager`, and sends them to the EventStore.
- class [DispatcherApp](#)
`DispatcherApp` is an `artdaq::Commandable` derived class which controls the `DispatcherCore`.
- class [DispatcherCore](#)
`DispatcherCore` implements the state machine for the Dispatcher artdaq application. `DispatcherCore` processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.
- class [EventBuilderApp](#)
`EventBuilderApp` is an `artdaq::Commandable` derived class which controls the `EventBuilderCore`.
- class [EventBuilderCore](#)
`EventBuilderCore` implements the state machine for the EventBuilder artdaq application. `EventBuilderCore` receives Fragment objects from the `DataReceiverManager`, and sends them to the EventStore.
- class [MPISentry](#)
The `MPISentry` class initializes and finalizes the MPI context that the artdaq applications run in.
- class [CapacityTestPolicy](#)
A `RoutingMasterPolicy` which tries to fully load the first receiver, then the second, and so on.
- class [NoOpPolicy](#)
A `RoutingMasterPolicy` which simply assigns Sequence IDs to tokens in the order they were received.
- class [RoundRobinPolicy](#)
A `RoutingMasterPolicy` which evenly distributes Sequence IDs to all receivers. If an uneven number of tokens have been received, extra tokens are stored for the next table update.
- class [RoutingMasterPolicy](#)
The interface through which `RoutingMasterCore` obtains Routing Tables using received Routing Tokens.
- class [RoutingMasterApp](#)
`RoutingMasterApp` is an `artdaq::Commandable` derived class which controls the `RoutingMasterCore` state machine.
- class [RoutingMasterCore](#)
`RoutingMasterCore` implements the state machine for the RoutingMaster artdaq application. `RoutingMasterCore` collects tokens from receivers, and at regular intervals uses these tokens to build Routing Tables that are sent to the senders.
- class [StatisticsHelper](#)

*This class manages MonitoredQuantity instances for the *Core classes.*

- class [BuildInfo](#)

BuildInfo is an art::EDProducer which saves information about package builds to the data file.

- class [TransferWrapper](#)

TransferWrapper wraps a TransferInterface so that it can be used in the ArtdaqInput class to make an art::Source.

- class [EventDump](#)

Write Event information to the console.

- class [PrintBuildInfo](#)

An art::EDAnalyzer which prints any artdaq::BuildInfo objects stored in the run.

- class [RandomDelayFilter](#)

A filter which delays for a random amount of time, then drops a random fraction of events. Used to simulate the delays and efficiency of real filters.

- struct [GetPackageBuildInfo](#)

Wrapper around the artdaq::GetPackageBuildInfo::getPackageBuildInfo function.

- class [GenericFragmentSimulator](#)

GenericFragmentSimulator creates simulated Generic events, with data distributed according to a "histogram" provided in the configuration data.

- class [Globals](#)

The artdaq::Globals class contains several variables which are useful across the entire artdaq system.

- struct [NetMonHeader](#)

Header with length information for NetMonTransport messages.

- class [DataReceiverManager](#)

Receives Fragment objects from one or more DataSenderManager instances using TransferInterface plugins. Data-ReceiverMaanger runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

- class [DataSenderManager](#)

Sends Fragment objects using TransferInterface plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations.

- class [RequestSender](#)

The RequestSender contains methods used to send data requests and Routing tokens.

- class [art_config_file](#)

art_config_file wraps a temporary file used to configure art

- class [SharedMemoryEventManager](#)

The SharedMemoryEventManager is a SharedMemoryManger which tracks events as they are built.

- class [CommanderInterface](#)

This interface defines the functions used to transfer data between artdaq applications.

- class [cmd_](#)

The "cmd_" class serves as the base class for all artdaq's XML-RPC commands.

- class [init_](#)

- class [soft_init_](#)

- class [reinit_](#)

- class [start_](#)

Command class representing a start transition.

- class [pause_](#)

- class [resume_](#)

- class [stop_](#)

- class [shutdown_](#)

shutdown_ Command class

- class [status_](#)

- class [status_](#) *Command class*
- class [report_](#)
 - report_ Command class*
- class [legal_commands_](#)
 - legal_commands_ Command class*
- class [register_monitor_](#)
 - register_monitor_ Command class*
- class [unregister_monitor_](#)
 - unregister_monitor_ Command class*
- class [xmlrpc_commander](#)

The [xmlrpc_commander](#) class serves as the XMLRPC server run in each artdaq application.

- class [RTIDDS](#)
 - DDS Transport Implementation.*
- class [AutodetectTransfer](#)

The [AutodetectTransfer TransferInterface](#) plugin sets up a [Shmem_transfer](#) plugin or [TCPSocket_transfer](#) plugin depending if the source and destination are on the same host, to maximize throughput.

- class [MPITransfer](#)
 - [MPITransfer](#) is a [TransferInterface](#) implementation plugin that transfers data using MPI.*
- class [MulticastTransfer](#)

[MulticastTransfer](#) is a [TransferInterface](#) implementation plugin that transfers data using Multicast.

- class [NullTransfer](#)
 - [NullTransfer](#) does not send or receive data, but acts as if it did.*

- class [RTIDDSTransfer](#)

[RTIDDSTransfer](#) is a [TransferInterface](#) implementation plugin that transfers data using RTI DDS.

- class [ShmemTransfer](#)
 - A [TransferInterface](#) implementation plugin that transfers data using Shared Memory.*
- class [TCPSocketTransfer](#)

[TCPSocketTransfer](#) is a [TransferInterface](#) implementation plugin that sends data using TCP sockets.

- class [TransferInterface](#)
 - This interface defines the functions used to transfer data between artdaq applications.*
- class [FragmentReceiverManager](#)

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins. Data-ReceiverMaanger runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

- class [FragmentStoreElement](#)
 - This class contains tracking information for all Fragment objects which have been received from a specific source.*
- class [TransferTest](#)

Test a set of [TransferInterface](#) plugins.

- class [FragmentSniffer](#)
 - This art::EDAnalyzer plugin tries to get Fragments from each event, asserting that the correct number of Fragments were present.*

Typedefs

- `typedef std::unique_ptr< CommandableFragmentGenerator > makeFunc_t (fhicl::ParameterSet const &ps)`
Constructs a [CommandableFragmentGenerator](#) instance, and returns a pointer to it.
- `typedef artdaq::BuildInfo <&instanceName, artdaqcore::GetPackageBuildInfo, artdaq::GetPackageBuildInfo > ArtdaqBuildInfo`
Specialized [artdaq::BuildInfo](#) object for the Artdaq build info.
- `typedef art::Source < detail::SharedMemoryReader < artdaq::Fragment::MakeSystemTypeMap > > RawInput`
RawInput is a typedef of art::Source<detail::SharedMemoryReader>

Enumerations

- `enum RequestMode { Single, Buffer, Window, Ignored }`

The RequestMode enumeration contains the possible ways which CommandableFragmentGenerator responds to data requests.

Functions

- `std::unique_ptr< CommandableFragmentGenerator > makeCommandableFragmentGenerator (std::string const &generator_plugin_spec, fhicl::ParameterSet const &ps)`
Load a [CommandableFragmentGenerator](#) plugin.
- `std::unique_ptr< RoutingMasterPolicy > makeRoutingMasterPolicy (std::string const &policy_plugin_spec, fhicl::ParameterSet const &ps)`
Load a [RoutingMasterPolicy](#) plugin.
- `std::string infoFilename (std::string const &prefix, int rank, int run)`
Generate a filename using the given parameters.
- `std::unique_ptr< artdaq::CommanderInterface > MakeCommanderPlugin (const fhicl::ParameterSet &pset, artdaq::Commandable &commandable)`
Load a [CommanderInterface](#) plugin.
- `std::string exception_msg (const std::runtime_error &er, const std::string &helpText="execute request")`
Write an exception message.
- `std::string exception_msg (const art::Exception &er, const std::string &helpText)`
Write an exception message.
- `std::string exception_msg (const cet::exception &er, const std::string &helpText)`
Write an exception message.
- `std::string exception_msg (const std::string &erText, const std::string &helpText)`
Write an exception message.
- `std::string exception_msg (const std::string &erText, const std::string &helpText)`
Write an exception message.
- `template<> std::string cmd_::getParam< std::string > (const xmlrpc_c::paramList ¶mList, int index)`
Get a parameter from the parameter list.

- template<>
art::RunID [cmd_::getParam](#)< art::RunID > (const xmlrpc_c::paramList ¶mList, int index)
Get a parameter from the parameter list.
- template<>
fhicl::ParameterSet [cmd_::getParam](#)< fhicl::ParameterSet > (const xmlrpc_c::paramList ¶mList, int index)
Get a parameter from the parameter list.
- std::unique_ptr
< [artdaq::TransferInterface](#) > [MakeTransferPlugin](#) (const fhicl::ParameterSet &pset, std::string plugin_label,
[TransferInterface::Role](#) role)
Load a [TransferInterface](#) plugin.

Variables

- static std::string [instanceName](#) = "ArtdaqBuildInfo"
Name of this [BuildInfo](#) instance.

6.6.1 Detailed Description

The artdaq namespace. Namespace used to differentiate the artdaq version of [GetPackageBuildInfo](#) from other versions present in the system.

6.6.2 Typedef Documentation

6.6.2.1 [typedef std::unique_ptr< artdaq::RoutingMasterPolicy > artdaq::makeFunc_t\(fhicl::ParameterSet const &ps\)](#)

Constructs a [CommandableFragmentGenerator](#) instance, and returns a pointer to it.

Constructs a [RoutingMasterPolicy](#) instance, and returns a pointer to it.

Parameters

<i>ps</i>	Parameter set for initializing the CommandableFragmentGenerator
-----------	---

Returns

A smart pointer to the [CommandableFragmentGenerator](#)

Parameters

<i>ps</i>	Parameter set for initializing the RoutingMasterPolicy
-----------	--

Returns

A smart pointer to the [RoutingMasterPolicy](#)

Definition at line 16 of file GeneratorMacros.hh.

6.6.3 Function Documentation

6.6.3.1 [template<> art::RunID artdaq::cmd_::getParam< art::RunID > \(const xmlrpc_c::paramList & paramList, int index \)](#)

Get a parameter from the parameter list.

Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

Returns

The requested parameter

This specialized cmd_getParam for the art::RunID type

Definition at line 240 of file xmlrpc_commander.cc.

6.6.3.2 template<> fhicl::ParameterSet artdaq::cmd_::getParam< fhicl::ParameterSet > (const xmlrpc_c::paramList & *paramList*, int *index*)

Get a parameter from the parameter list.

Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

Returns

The requested parameter

This specialized cmd_getParam for the fhicl::ParameterSet type

Definition at line 259 of file xmlrpc_commander.cc.

6.6.3.3 template<> std::string artdaq::cmd_::getParam< std::string > (const xmlrpc_c::paramList & *paramList*, int *index*)

Get a parameter from the parameter list.

Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

Returns

The requested parameter

This specialized cmd_getParam for the std::string type

Definition at line 226 of file xmlrpc_commander.cc.

6.6.3.4 std::string artdaq::exception_msg (const std::runtime_error & *er*, const std::string & *helpText* = "execute request")

Write an exception message.

Parameters

<i>er</i>	A std::runtime_error to print
<i>helpText</i>	Additional information about the exception context. Default: "execute request"

Returns

Exception message

Definition at line 40 of file xmlrpc_commander.cc.

6.6.3.5 std::string artdaq::exception_msg (const art::Exception & er, const std::string & helpText)

Write an exception message.

Parameters

<i>er</i>	An art::Exception to print
<i>helpText</i>	Additional information abou the exception context

Returns

Exception message

Definition at line 57 of file xmlrpc_commander.cc.

6.6.3.6 std::string artdaq::exception_msg (const cet::exception & er, const std::string & helpText)

Write an exception message.

Parameters

<i>er</i>	A cet::exceptio to print
<i>helpText</i>	Additional information abou the exception context

Returns

Exception message

Definition at line 74 of file xmlrpc_commander.cc.

6.6.3.7 std::string artdaq::exception_msg (const std::string & erText, const std::string & helpText)

Write an exception message.

Parameters

<i>erText</i>	A std::string to print
<i>helpText</i>	Additional information abou the exception context

Returns

Exception message

Definition at line 91 of file xmlrpc_commander.cc.

6.6.3.8 `std::string artdaq::infoFilename(std::string const & prefix, int rank, int run)`

Generate a filename using the given parameters.

Parameters

<i>prefix</i>	Prefix for the file name
<i>rank</i>	Rank of the application
<i>run</i>	Run number

Returns

prefix + *run* (4 digits) + "_" + *rank* (4 digits) + ".txt";

Definition at line 7 of file infoFilename.cc.

6.6.3.9 `std::unique_ptr< artdaq::CommandableFragmentGenerator > artdaq::makeCommandableFragmentGenerator (std::string const & generator_plugin_spec, fhicl::ParameterSet const & ps)`

Load a [CommandableFragmentGenerator](#) plugin.

Parameters

<i>generator_plugin-spec</i>	Name of the plugin
<i>ps</i>	ParameterSet used to configure the plugin

Returns

Pointer to the new plugin instance

Definition at line 8 of file makeCommandableFragmentGenerator.cc.

6.6.3.10 `std::unique_ptr< CommanderInterface > artdaq::MakeCommanderPlugin (const fhicl::ParameterSet & pset, artdaq::Commandable & commandable)`

Load a [CommanderInterface](#) plugin.

Parameters

<i>pset</i>	ParameterSet used to configure the CommanderInterface
<i>commandable</i>	artdaq::Commandable object to send transition commands to

Returns

Pointer to the new [CommanderInterface](#) instance

Definition at line 12 of file MakeCommanderPlugin.cc.

6.6.3.11 `std::unique_ptr< artdaq::RoutingMasterPolicy > artdaq::makeRoutingMasterPolicy (std::string const & policy_plugin_spec, fhicl::ParameterSet const & ps)`

Load a [RoutingMasterPolicy](#) plugin.

Parameters

<i>policy_plugin_- spec</i>	Name of the RoutingMasterPolicy
<i>ps</i>	ParameterSet used to configure the RoutingMasterPolicy

Returns

`std::unique_ptr<RoutingMasterPolicy>` to the new [RoutingMasterPolicy](#) instance

Definition at line 8 of file `makeRoutingMasterPolicy.cc`.

6.6.3.12 `std::unique_ptr< TransferInterface > artdaq::MakeTransferPlugin (const fhicl::ParameterSet & pset, std::string plugin_label, TransferInterface::Role role)`

Load a [TransferInterface](#) plugin.

Parameters

<i>pset</i>	ParameterSet used to configure the TransferInterface
<i>plugin_label</i>	Name of the plugin
<i>role</i>	Whether the TransferInterface should be configured as kSend or kReceive

Returns

Pointer to the new [TransferInterface](#) instance

Definition at line 12 of file `MakeTransferPlugin.cc`.

6.7 artdaq::detail Namespace Reference

The [ardaq::detail](#) namespace contains internal implementation details for some classes.

Classes

- struct [SharedMemoryReader](#)

The SharedMemoryReader is a class which implements the methods needed by art::Source.
- class [FragCounter](#)

Keep track of the count of Fragments received from a set of sources.
- struct [RequestPacket](#)

The RequestPacket contains information about a single data request.
- struct [RequestHeader](#)

Header of a RequestMessage. Contains magic bytes for validation and a count of expected RequestPackets.
- class [RequestMessage](#)

A RequestMessage consists of a RequestHeader and zero or more RequestPackets. They will usually be sent in two calls to send()
- struct [RoutingPacketEntry](#)

A row of the Routing Table.
- struct [RoutingPacketHeader](#)

The header of the Routing Table, containing the magic bytes and the number of entries.

- struct [RoutingAckPacket](#)

A [RoutingAckPacket](#) contains the rank of the table receiver, plus the first and last sequence IDs in the Routing Table (for verification)

- struct [RoutingToken](#)

The [RoutingToken](#) contains the magic bytes, the rank of the token sender, and the number of slots free. This is a TCP message, so additional verification is not necessary.

Typedefs

- using [RoutingPacket](#) = std::vector< [RoutingPacketEntry](#) >

A [RoutingPacket](#) is simply a vector of [RoutingPacketEntry](#) objects. It is not suitable for network transmission, rather a [RoutingPacketHeader](#) should be sent, followed by &[RoutingPacket.at\(0\)](#) (the physical storage of the vector)

Enumerations

- enum [TaskType](#) : int { **BoardReaderTask** =1, **EventBuilderTask** =2, **AggregatorTask** =3, **RoutingMasterTask** =4 }

The types of applications in artdaq.

- enum [RequestMessageMode](#) : uint8_t { [RequestMessageMode::Normal](#) = 0, [RequestMessageMode::EndOfRun](#) = 1 }

Mode used to indicate current run conditions to the request receiver.

- enum [RoutingMasterMode](#) : uint8_t { [RoutingMasterMode::RouteBySequenceID](#), [RoutingMasterMode::RouteBySendCount](#), **INVALID** }

Mode indicating whether the RoutingMaster is routing events by Sequence ID or by Send Count.

Functions

- std::ostream & [operator<<](#) (std::ostream &o, [RequestMessageMode](#) m)

Converts the RequestMessageMode to a string and sends it to the output stream.

6.7.1 Detailed Description

The [artdaq::detail](#) namespace contains internal implementation details for some classes.

6.7.2 Enumeration Type Documentation

6.7.2.1 enum [artdaq::detail::RequestMessageMode](#) : uint8_t [strong]

Mode used to indicate current run conditions to the request receiver.

Enumerator

Normal Normal running.

EndOfRun End of Run mode (Used to end request processing on receiver)

Definition at line 17 of file RequestMessage.hh.

6.7.2.2 enum artdaq::detail::RoutingMasterMode : uint8_t [strong]

Mode indicating whether the RoutingMaster is routing events by Sequence ID or by Send Count.

Enumerator

RouteBySequenceID Events should be routed by sequence ID (BR -> EB)

RouteBySendCount Events should be routed by send count (EB -> Agg)

Definition at line 24 of file RoutingPacket.hh.

6.7.3 Function Documentation**6.7.3.1 std::ostream& artdaq::detail::operator<< (std::ostream & o, RequestMessageMode m) [inline]**

Converts the RequestMessageMode to a string and sends it to the output stream.

Parameters

<i>o</i>	Stream to send string to
<i>m</i>	RequestMessageMode to convert to string

Returns

o with string sent to it

Definition at line 29 of file RequestMessage.hh.

Chapter 7

Class Documentation

7.1 artdaq::art_config_file Class Reference

[art_config_file](#) wraps a temporary file used to configure art

```
#include <ardaq/DAQrate/SharedMemoryEventManager.hh>
```

Public Member Functions

- [art_config_file](#) (fhicl::ParameterSet ps)
art_config_file Constructor
- std::string [getFileName](#) () const
Get the path of the temporary file.

7.1.1 Detailed Description

[art_config_file](#) wraps a temporary file used to configure art

Definition at line 19 of file SharedMemoryEventManager.hh.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 artdaq::art_config_file::art_config_file (fhicl::ParameterSet ps) [inline]

[art_config_file](#) Constructor

Parameters

<i>ps</i>	ParameterSet to write to temporary file
-----------	---

Definition at line 26 of file SharedMemoryEventManager.hh.

7.1.3 Member Function Documentation

7.1.3.1 std::string artdaq::art_config_file::getFileName() const [inline]

Get the path of the temporary file.

Returns

The path of the temporary file

Definition at line 51 of file SharedMemoryEventManager.hh.

The documentation for this class was generated from the following file:

- artdaq/artdaq/DAQrate/SharedMemoryEventManager.hh

7.2 art::ArtdaqInput< U > Class Template Reference

This template class provides a unified interface for reading data into art.

```
#include <artdaq/ArtModules/ArtdaqInput.hh>
```

Public Member Functions

- `ArtdaqInput` (const `ArtdaqInput` &)=delete
Copy Constructor is deleted.
- `ArtdaqInput & operator=` (const `ArtdaqInput` &)=delete
Copy Assignment operator is deleted.
- `~ArtdaqInput` ()
ArtdaqInput Destructor.
- `ArtdaqInput` (const `fhicl::ParameterSet` &ps, `art::ProductRegistryHelper` &helper, const `art::SourceHelper` &pm)
ArtdaqInput Constructor.
- `void closeCurrentFile` ()
Called by art to close the input source. No-Op.
- `void readFile` (const `std::string` &, `art::FileBlock` *&fb)
Emulate reading a file.
- `bool hasMoreData` () const
Whether additional events are expected from the source.
- `bool readNext` (`art::RunPrincipal` *const inR, `art::SubRunPrincipal` *const inSR, `art::RunPrincipal` *&outR, `art::SubRunPrincipal` *&outSR, `art::EventPrincipal` *&outE)
Read the next event from the communication wrapper.

7.2.1 Detailed Description

```
template<typename U>class art::ArtdaqInput< U >
```

This template class provides a unified interface for reading data into art.

Template Parameters

<i>U</i>	The class responsible for delivering data
----------	---

JCF, May-27-2016 [ArtdaqInput](#) is a template class which takes, as a parameter, a class which it uses to receive data; the instance of this class is called "communicationWrapper_". As of this writing, this wrapper class is implemented by [NetMonWrapper](#) (for reading data into the aggregator from the eventbuilder) and [TransferWrapper](#) (for reading data into an art process). This class presents a unified approach to handling art provenance, regardless of the communication protocol used to read data in.

Definition at line 57 of file ArtdaqInput.hh.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 `template<typename U > art::ArtdaqInput< U >::ArtdaqInput (const fhicl::ParameterSet & ps,
art::ProductRegistryHelper & helper, const art::SourceHelper & pm)`

[ArtdaqInput](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configre communication wrapper class
<i>helper</i>	An art::ProductRegistryHelper for registering products
<i>pm</i>	An art::SourceHelper for handling provenance

Definition at line 162 of file ArtdaqInput.hh.

7.2.3 Member Function Documentation

7.2.3.1 `template<typename U > bool art::ArtdaqInput< U >::hasMoreData () const`

Whether additional events are expected from the source.

Returns

True if [ArtdaqInput](#) has not been shut down

Definition at line 309 of file ArtdaqInput.hh.

7.2.3.2 `template<typename U > ArtdaqInput& art::ArtdaqInput< U >::operator= (const ArtdaqInput< U > &)
[delete]`

Copy Assignment operator is deleted.

Returns

[ArtdaqInput](#) copy

7.2.3.3 `template<typename U > void art::ArtdaqInput< U >::readFile (const std::string & , art::FileBlock *& fb)`

Emulate reading a file.

Parameters

<i>fb</i>	Output art::FileBlock object
-----------	------------------------------

Definition at line 296 of file ArtdaqInput.hh.

7.2.3.4 template<typename U > bool art::ArtdaqInput< U >::readNext (art::RunPrincipal *const *inR*, art::SubRunPrincipal *const *inSR*, art::RunPrincipal *& *outR*, art::SubRunPrincipal *& *outSR*, art::EventPrincipal *& *outE*)

Read the next event from the communication wrapper.

Parameters

<i>inR</i>	RunPrincipal input pointer
<i>inSR</i>	SubRunPrincipal input pointer
<i>outR</i>	RunPrincipal output pointer
<i>outSR</i>	SubRunPrincipal output pointer
<i>outE</i>	EventPrincipal output pointer

Returns

Whether an event was successfully read from the communication wrapper

Definition at line 577 of file ArtdaqInput.hh.

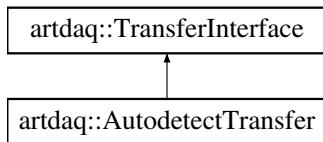
The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/ArtdaqInput.hh

7.3 artdaq::AutodetectTransfer Class Reference

The [AutodetectTransfer TransferInterface](#) plugin sets up a Shmem_transfer plugin or TCPSocket_transfer plugin depending if the source and destination are on the same host, to maximize throughput.

Inheritance diagram for artdaq::AutodetectTransfer:



Public Member Functions

- [AutodetectTransfer](#) (const fhicl::ParameterSet &pset, [Role role](#))

AutodetectTransfer Constructor.
- virtual [~AutodetectTransfer](#) ()=default

AutodetectTransfer default Destructor.
- int [receiveFragment](#) (artdaq::Fragment &fragment, size_t receiveTimeout) override

Receive a Fragment, using the underlying transfer plugin.
- int [receiveFragmentHeader](#) (detail::RawFragmentHeader &header, size_t receiveTimeout) override

Receive a Fragment Header from the transport mechanism.

- int `receiveFragmentData` (RawDataType *destination, size_t wordCount) override

Receive the body of a Fragment to the given destination pointer.

- CopyStatus `copyFragment` (artdaq::Fragment &fragment, size_t send_timeout_usec=std::numeric_limits<size_t>::max()) override

Send a Fragment in non-reliable mode, using the underlying transfer plugin.

- CopyStatus `moveFragment` (artdaq::Fragment &&fragment, size_t send_timeout_usec=std::numeric_limits<size_t>::max()) override

Send a Fragment in reliable mode, using the underlying transfer plugin.

Additional Inherited Members

7.3.1 Detailed Description

The [AutodetectTransfer TransferInterface](#) plugin sets up a Shmem_transfer plugin or TCPSocket_transfer plugin depending if the source and destination are on the same host, to maximize throughput.

Definition at line 13 of file Autodetect_transfer.cc.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 artdaq::AutodetectTransfer::AutodetectTransfer (const fhicl::ParameterSet & pset, Role role)

[AutodetectTransfer](#) Constructor.

Parameters

<code>pset</code>	ParameterSet used to configure AutodetectTransfer
<code>role</code>	Role of this TransferInterface , either kReceive or kSend

Definition at line 91 of file Autodetect_transfer.cc.

7.3.3 Member Function Documentation

7.3.3.1 CopyStatus artdaq::AutodetectTransfer::copyFragment (artdaq::Fragment & fragment, size_t send_timeout_usec = std::numeric_limits<size_t>::max()) [inline], [override], [virtual]

Send a Fragment in non-reliable mode, using the underlying transfer plugin.

Parameters

<code>fragment</code>	The Fragment to send
<code>send_timeout_usec</code>	How long to wait before aborting. Defaults to size_t::MAX_VALUE

Returns

A [TransferInterface::CopyStatus](#) result variable

Implements [artdaq::TransferInterface](#).

Definition at line 68 of file Autodetect_transfer.cc.

7.3.3.2 **CopyStatus artdaq::AutodetectTransfer::moveFragment (artdaq::Fragment && *fragment*, size_t *send_timeout_usec* = std::numeric_limits<size_t>::max())** [inline], [override], [virtual]

Send a Fragment in reliable mode, using the underlying transfer plugin.

Parameters

<i>fragment</i>	The Fragment to send
<i>send_timeout_usec</i>	How long to wait before aborting. Defaults to size_t::MAX_VALUE

Returns

A [TransferInterface::CopyStatus](#) result variable

Implements [artdaq::TransferInterface](#).

Definition at line 80 of file Autodetect_transfer.cc.

```
7.3.3.3 int artdaq::AutodetectTransfer::receiveFragment( artdaq::Fragment & fragment, size_t receiveTimeout ) [inline],  
[override], [virtual]
```

Receive a Fragment, using the underlying transfer plugin.

Parameters

<i>fragment</i>	Output Fragment
<i>receiveTimeout</i>	Time to wait before returning TransferInterface::RECV_TIMEOUT

Returns

Rank of sender

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 34 of file Autodetect_transfer.cc.

```
7.3.3.4 int artdaq::AutodetectTransfer::receiveFragmentData( RawDataType * destination, size_t wordCount ) [inline],  
[override], [virtual]
```

Receive the body of a Fragment to the given destination pointer.

Parameters

<i>destination</i>	Pointer to memory region where Fragment data should be stored
<i>wordCount</i>	Number of words of Fragment data to receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implements [artdaq::TransferInterface](#).

Definition at line 57 of file Autodetect_transfer.cc.

```
7.3.3.5 int artdaq::AutodetectTransfer::receiveFragmentHeader( detail::RawFragmentHeader & header, size_t receiveTimeout )  
[inline], [override], [virtual]
```

Receive a Fragment Header from the transport mechanism.

Parameters

<code>out</code>	<code>header</code>	Received Fragment Header
	<code>receiveTimeout</code>	Timeout for receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implements [artdaq::TransferInterface](#).

Definition at line 46 of file Autodetect_transfer.cc.

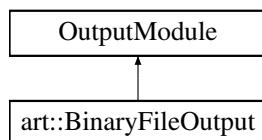
The documentation for this class was generated from the following file:

- artdaq/artdaq/TransferPlugins/Autodetect_transfer.cc

7.4 art::BinaryFileOutput Class Reference

The [BinaryFileOutput](#) module streams art Events to a binary file, bypassing ROOT.

Inheritance diagram for art::BinaryFileOutput:



Public Member Functions

- `BinaryFileOutput (ParameterSet const &ps)`
BinaryFileOutput Constructor.
- virtual `~BinaryFileOutput ()`
BinaryFileOutput Destructor.

7.4.1 Detailed Description

The [BinaryFileOutput](#) module streams art Events to a binary file, bypassing ROOT.

Definition at line 40 of file BinaryFileOutput_module.cc.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 art::BinaryFileOutput::BinaryFileOutput (ParameterSet const & ps) [explicit]

[BinaryFileOutput](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure BinaryFileOutput
-----------------	---

[BinaryFileOutput](#) accepts the same configuration parameters as `art::OutputModule`. It has the same name substitution code that `RootOutput` uses to uniquify names.

[BinaryFileOutput](#) also expects the following Parameters: "fileName" (REQUIRED): Name of the file to write "directIO" (Default: false): Whether to use O_DIRECT

Definition at line 87 of file `BinaryFileOutput_module.cc`.

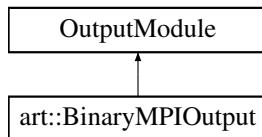
The documentation for this class was generated from the following file:

- `artdaq/artdaq/ArtModules/BinaryFileOutput_module.cc`

7.5 art::BinaryMPIOutput Class Reference

An `art::OutputModule` which sends Fragments using `DataSenderManager`. This module produces output identical to that of a `BoardReader`, for use in systems which have multiple layers of `EventBuilders`.

Inheritance diagram for `art::BinaryMPIOutput`:



Public Member Functions

- `BinaryMPIOutput (ParameterSet const &ps)`
BinaryMPIOutput Constructor.
- `virtual ~BinaryMPIOutput ()`
BinaryMPIOutput Destructor.

7.5.1 Detailed Description

An `art::OutputModule` which sends Fragments using `DataSenderManager`. This module produces output identical to that of a `BoardReader`, for use in systems which have multiple layers of `EventBuilders`.

Definition at line 39 of file `BinaryMPIOutput_module.cc`.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 art::BinaryMPIOutput::BinaryMPIOutput (ParameterSet const & ps) [explicit]

[BinaryMPIOutput](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure BinaryMPIOOutput
-----------------	---

[BinaryMPIOOutput](#) forwards its ParameterSet to `art::OutputModule`, so any Parameters it requires are also required by [BinaryMPIOOutput](#). [BinaryMPIOOutput](#) also forwards its ParameterSet to `DataSenderManager`, so any Parameters it requires are *also* required by [BinaryMPIOOutput](#). Finally, [BinaryMPIOOutput](#) accepts the following parameters:

- "rt_-priority" (Default: 0): Priority for this thread "module_name" (Default: [BinaryMPIOOutput](#)): Friendly name for this module (MessageFacility Category)

Definition at line 85 of file `BinaryMPIOOutput_module.cc`.

The documentation for this class was generated from the following file:

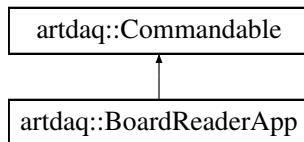
- `artdaq/artdaq/ArtModules/BinaryMPIOOutput_module.cc`

7.6 `artdaq::BoardReaderApp` Class Reference

[BoardReaderApp](#) is an `artdaq::Commandable` derived class which controls the [BoardReaderCore](#) state machine.

```
#include <artdaq/Application/BoardReaderApp.hh>
```

Inheritance diagram for `artdaq::BoardReaderApp`:



Public Member Functions

- `BoardReaderApp (int rank, std::string name)`
BoardReaderApp Constructor.
- `BoardReaderApp (BoardReaderApp const &) = delete`
Copy Constructor is deleted.
- `virtual ~BoardReaderApp () = default`
Default Destructor.
- `BoardReaderApp & operator= (BoardReaderApp const &) = delete`
Copy Assignment Operator is deleted.
- `bool do_initialize (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp) override`
Initialize the BoardReaderCore.
- `bool do_start (art::RunID id, uint64_t timeout, uint64_t timestamp) override`
Start the BoardReaderCore.
- `bool do_stop (uint64_t timeout, uint64_t timestamp) override`
Stop the BoardReaderCore.
- `bool do_pause (uint64_t timeout, uint64_t timestamp) override`
Pause the BoardReaderCore.
- `bool do_resume (uint64_t timeout, uint64_t timestamp) override`
Resume the BoardReaderCore.
- `bool do_shutdown (uint64_t timeout) override`

Shutdown the BoardReaderCore.

- bool `do_soft_initialize` (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp) override

Soft-Initialize the BoardReaderCore.

- bool `do_reinitialize` (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp) override

Reinitialize the BoardReaderCore.

- void `BootedEnter` () override

Action taken upon entering the "Booted" state.

- std::string `report` (std::string const &which) const override

If which is "transition_status", report the status of the last transition. Otherwise pass through to BoardReaderCore.

Additional Inherited Members

7.6.1 Detailed Description

`BoardReaderApp` is an `artdaq::Commandable` derived class which controls the `BoardReaderCore` state machine.

Definition at line 18 of file `BoardReaderApp.hh`.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 artdaq::BoardReaderApp::BoardReaderApp (int rank, std::string name)

`BoardReaderApp` Constructor.

Parameters

<code>rank</code>	Rank of this BoardReader
<code>name</code>	Friendly name of this application instance (MessageFacility Category)

Definition at line 3 of file `BoardReaderApp.cc`.

7.6.3 Member Function Documentation

7.6.3.1 void artdaq::BoardReaderApp::BootedEnter () [override], [virtual]

Action taken upon entering the "Booted" state.

This resets the `BoardReaderCore` pointer

Reimplemented from `artdaq::Commandable`.

Definition at line 154 of file `BoardReaderApp.cc`.

7.6.3.2 bool artdaq::BoardReaderApp::do_initialize (fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp) [override], [virtual]

Initialize the `BoardReaderCore`.

Parameters

<i>pset</i>	ParameterSet used to configure the BoardReaderCore
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 11 of file BoardReaderApp.cc.

7.6.3.3 bool artdaq::BoardReaderApp::do_pause (*uint64_t timeout, uint64_t timestamp*) [override], [virtual]

Pause the [BoardReaderCore](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 79 of file BoardReaderApp.cc.

7.6.3.4 bool artdaq::BoardReaderApp::do_reinitialize (*fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp*) [override], [virtual]

Reinitialize the [BoardReaderCore](#).

Parameters

<i>pset</i>	ParameterSet used to configure the BoardReaderCore
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 142 of file BoardReaderApp.cc.

7.6.3.5 bool artdaq::BoardReaderApp::do_resume (*uint64_t timeout, uint64_t timestamp*) [override], [virtual]

Resume the [BoardReaderCore](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 100 of file BoardReaderApp.cc.

7.6.3.6 bool artdaq::BoardReaderApp::do_shutdown (uint64_t *timeout*) [override], [virtual]

Shutdown the [BoardReaderCore](#).

Parameters

<i>timeout</i>	Timeout for transition
----------------	------------------------

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 117 of file BoardReaderApp.cc.

7.6.3.7 bool artdaq::BoardReaderApp::do_soft_initialize (fhicl::ParameterSet const & *pset*, uint64_t *timeout*, uint64_t *timestamp*) [override], [virtual]

Soft-Initialize the [BoardReaderCore](#).

Parameters

<i>pset</i>	ParameterSet used to configure the BoardReaderCore
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 129 of file BoardReaderApp.cc.

7.6.3.8 bool artdaq::BoardReaderApp::do_start (art::RunID *id*, uint64_t *timeout*, uint64_t *timestamp*) [override], [virtual]

Start the [BoardReaderCore](#).

Parameters

<i>id</i>	Run ID of new run
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 33 of file BoardReaderApp.cc.

7.6.3.9 bool artdaq::BoardReaderApp::do_stop (uint64_t *timeout*, uint64_t *timestamp*) [override], [virtual]

Stop the [BoardReaderCore](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 57 of file BoardReaderApp.cc.

7.6.3.10 BoardReaderApp& artdaq::BoardReaderApp::operator= (BoardReaderApp const &) [delete]

Copy Assignment Operator is deleted.

Returns

[BoardReaderApp](#) copy

7.6.3.11 std::string artdaq::BoardReaderApp::report (std::string const & *which*) const [override], [virtual]

If *which* is "transition_status", report the status of the last transition. Otherwise pass through to [BoardReaderCore](#).

Parameters

<i>which</i>	What to report on
--------------	-------------------

Returns

Report string. Empty for unknown "which" parameter

Reimplemented from [artdaq::Commandable](#).

Definition at line 165 of file BoardReaderApp.cc.

The documentation for this class was generated from the following files:

- artdaq/artdaq/Application/BoardReaderApp.hh
- artdaq/artdaq/Application/BoardReaderApp.cc

7.7 artdaq::BoardReaderCore Class Reference

[BoardReaderCore](#) implements the state machine for the BoardReader artdaq application. It contains a [CommandableFragmentGenerator](#), which generates Fragments which are then sent to a [DataSenderManager](#) by [BoardReaderCore](#).

```
#include <artdaq/Application/BoardReaderCore.hh>
```

Public Member Functions

- [BoardReaderCore \(Commandable &parent_application, int rank, std::string name\)](#)
BoardReaderCore Constructor.
- [BoardReaderCore \(BoardReaderCore const &\)=delete](#)
Copy Constructor is Deleted.
- [virtual ~BoardReaderCore \(\)](#)
BoardReaderCore Destructor.
- [BoardReaderCore & operator= \(BoardReaderCore const &\)=delete](#)
Copy Assignment Operator is deleted.
- [bool initialize \(fhicl::ParameterSet const &pset, uint64_t, uint64_t\)](#)
Initialize the BoardReaderCore.
- [bool start \(art::RunID id, uint64_t timeout, uint64_t timestamp\)](#)
Start the BoardReader, and the CommandableFragmentGenerator.
- [bool stop \(uint64_t timeout, uint64_t timestamp\)](#)
Stop the BoardReader, and the CommandableFragmentGenerator.
- [bool pause \(uint64_t timeout, uint64_t timestamp\)](#)
Pause the BoardReader, and the CommandableFragmentGenerator.
- [bool resume \(uint64_t timeout, uint64_t timestamp\)](#)
Resume the BoardReader, and the CommandableFragmentGenerator.
- [bool shutdown \(uint64_t\)](#)
Shutdown the BoardReader, and the CommandableFragmentGenerator.
- [bool soft_initialize \(fhicl::ParameterSet const &pset, uint64_t, uint64_t\)](#)
Soft-Initialize the BoardReader. No-Op.
- [bool reinitialize \(fhicl::ParameterSet const &pset, uint64_t, uint64_t\)](#)
Reinitialize the BoardReader. No-Op.
- [size_t process_fragments \(\)](#)
Main working loop of the BoardReaderCore.
- [std::string report \(std::string const &which\) const](#)
Send a report on a given run-time quantity.

Static Public Member Functions

- static [DataSenderManager * GetDataSenderManagerPtr \(\)](#)
Gets a handle to the DataSenderManager.

Static Public Attributes

- static const std::string **FRAGMENTS_PROCESSED_STAT_KEY**
Key for the Fragments Processed MonitoredQuantity.
- static const std::string **INPUT_WAIT_STAT_KEY**
Key for the Input Wait MonitoredQuantity.
- static const std::string **BRSYNC_WAIT_STAT_KEY**
Key for the Sync Wait MonitoredQuantity.
- static const std::string **OUTPUT_WAIT_STAT_KEY**
Key for the Output Wait MonitoredQuantity.
- static const std::string **FRAGMENTS_PER_READ_STAT_KEY**
Key for the Fragments Per Read MonitoredQuantity.

7.7.1 Detailed Description

`BoardReaderCore` implements the state machine for the BoardReader artdaq application. It contains a `Commandable-FragmentGenerator`, which generates Fragments which are then sent to a `DataSenderManager` by `BoardReaderCore`.

Definition at line 23 of file `BoardReaderCore.hh`.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 `artdaq::BoardReaderCore::BoardReaderCore (Commandable & parent_application, int rank, std::string name)`

`BoardReaderCore` Constructor.

Parameters

<code>parent_-application</code>	Reference to parent <code>Commandable</code> object, for <code>in_run_failure</code> notification
<code>rank</code>	Rank of the BoardReader
<code>name</code>	Friendly name for the BoardReader

Definition at line 27 of file `BoardReaderCore.cc`.

7.7.3 Member Function Documentation

7.7.3.1 `static DataSenderManager* artdaq::BoardReaderCore::GetDataSenderManagerPtr () [inline], [static]`

Gets a handle to the `DataSenderManager`.

Returns

Pointer to the `DataSenderManager`

Definition at line 151 of file `BoardReaderCore.hh`.

7.7.3.2 `bool artdaq::BoardReaderCore::initialize (fhicl::ParameterSet const & pset, uint64_t , uint64_t)`

Initialize the `BoardReaderCore`.

Parameters

<i>pset</i>	ParameterSet used to configure the BoardReaderCore
-------------	--

Returns

True if the initialize attempt succeeded

```
* BoardReaderCore accepts the following Parameters:
* "daq" (REQUIRED): FHiCL table containing DAQ configuration.
* "fragment_receiver" (REQUIRED): FHiCL table containing Fragment Receiver configuration.
* See CommandableFragmentGenerator for configuration options.
* "generator" (Default: ""): The plugin name of the generator to load
* "rt_priority" (Default: 0): The unix priority to attempt to assign to the process
* "metrics": FHiCL table containing MetricManager configuration.
* See MetricManager for configuration options.
```

Definition at line 51 of file BoardReaderCore.cc.

7.7.3.3 [BoardReaderCore& artdaq::BoardReaderCore::operator=\(BoardReaderCore const & \)](#) [delete]

Copy Assignment Operator is deleted.

Returns

[BoardReaderCore](#) copy

7.7.3.4 [bool artdaq::BoardReaderCore::pause\(uint64_t timeout, uint64_t timestamp \)](#)

Pause the BoardReader, and the [CommandableFragmentGenerator](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

True unless exception occurred

Definition at line 214 of file BoardReaderCore.cc.

7.7.3.5 [size_t artdaq::BoardReaderCore::process_fragments\(\)](#)

Main working loop of the [BoardReaderCore](#).

Returns

Number of Fragments generated

This loop calls the [CommandableFragmentGenerator::getNext](#) method, then sends each Fragment using [DataSenderManager](#).

Definition at line 256 of file BoardReaderCore.cc.

7.7.3.6 `bool artdaq::BoardReaderCore::reinitialize(fhicl::ParameterSet const & pset, uint64_t , uint64_t)`

Reinitialize the BoardReader. No-Op.

Parameters

<i>pset</i>	ParameterSet used to configure the BoardReaderCore
-------------	--

Returns

True unless exception occurred

Definition at line 248 of file [BoardReaderCore.cc](#).

7.7.3.7 std::string artdaq::BoardReaderCore::report (std::string const & *which*) const

Send a report on a given run-time quantity.

Parameters

<i>which</i>	Which quantity to report
--------------	--------------------------

Returns

A string containing the requested quantity.

If the [CommandableFragmentGenerator](#) has been initialized, [CommandableFragmentGenerator::report\(std::string const& which\)](#) will be called. Otherwise, the [BoardReaderCore](#) will return the current run number and an error message.

Definition at line 489 of file [BoardReaderCore.cc](#).

7.7.3.8 bool artdaq::BoardReaderCore::resume (uint64_t *timeout*, uint64_t *timestamp*)

Resume the BoardReader, and the [CommandableFragmentGenerator](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

True unless exception occurred

Definition at line 224 of file [BoardReaderCore.cc](#).

7.7.3.9 bool artdaq::BoardReaderCore::shutdown (uint64_t)

Shutdown the BoardReader, and the [CommandableFragmentGenerator](#).

Returns

True unless exception occurred

Definition at line 233 of file [BoardReaderCore.cc](#).

7.7.3.10 bool artdaq::BoardReaderCore::soft_initialize (fhicl::ParameterSet const & *pset*, uint64_t , uint64_t)

Soft-Initialize the BoardReader. No-Op.

Parameters

<i>pset</i>	ParameterSet used to configure the BoardReaderCore
-------------	--

Returns

True unless exception occurred

Definition at line 240 of file [BoardReaderCore.cc](#).

7.7.3.11 bool artdaq::BoardReaderCore::start (art::RunID *id*, uint64_t *timeout*, uint64_t *timestamp*)

Start the BoardReader, and the [CommandableFragmentGenerator](#).

Parameters

<i>id</i>	Run ID of new run
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

True unless exception occurred

Definition at line 186 of file [BoardReaderCore.cc](#).

7.7.3.12 bool artdaq::BoardReaderCore::stop (uint64_t *timeout*, uint64_t *timestamp*)

Stop the BoardReader, and the [CommandableFragmentGenerator](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

True unless exception occurred

Definition at line 204 of file [BoardReaderCore.cc](#).

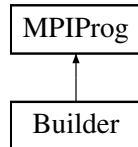
The documentation for this class was generated from the following files:

- artdaq/artdaq/Application/BoardReaderCore.hh
- artdaq/artdaq/Application/BoardReaderCore.cc

7.8 Builder Class Reference

The [Builder](#) class runs the builder test.

Inheritance diagram for Builder:



Public Member Functions

- **Builder** (int argc, char *argv[], fhicl::ParameterSet pset)

Builder Constructor.
- void **go** ()

*Start the **Builder** application, using the type configuration to select which method to run.*
- void **sink** ()

Receive data from source via DataReceiverManager, send it to the EventStore (and art, if configured)
- void **detector** ()

Generate data, and send it using DataSenderManager.

Additional Inherited Members

7.8.1 Detailed Description

The **Builder** class runs the builder test.

Definition at line 37 of file builder.cc.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Builder::Builder (int argc, char * argv[], fhicl::ParameterSet pset)

Builder Constructor.

Parameters

<i>argc</i>	Argument Count
<i>argv</i>	Argument Array
<i>pset</i>	fhicl::ParameterSet used to configure builder

Definition at line 79 of file builder.cc.

The documentation for this class was generated from the following file:

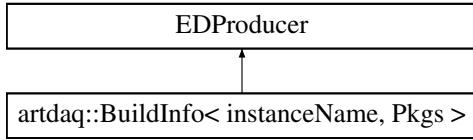
- artdaq/proto/builder.cc

7.9 artdaq::BuildInfo< instanceName, Pkgs > Class Template Reference

BuildInfo is an art::EDProducer which saves information about package builds to the data file.

```
#include <artdaq/ArtModules/BuildInfo_module.hh>
```

Inheritance diagram for artdaq::BuildInfo< instanceName, Pkgs >:



Public Member Functions

- `BuildInfo (fhicl::ParameterSet const &p)`
BuildInfo module Constructor.
- `virtual ~BuildInfo ()=default`
Default Destructor.
- `void beginRun (art::Run &r) override`
Perform actions at the beginning of the Run.
- `void produce (art::Event &e) override`
Perform actions for each event.

7.9.1 Detailed Description

`template<std::string * instanceName, typename... Pkgs>class artdaq::BuildInfo< instanceName, Pkgs >`

`BuildInfo` is an `art::EDProducer` which saves information about package builds to the data file.

Template Parameters

<code>instanceName</code>	Tag which the <code>BuildInfo</code> objects will be saved under
<code>Pkgs</code>	List of package <code>BuildInfo</code> types

Definition at line 19 of file `BuildInfo_module.hh`.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 `template<std::string * instanceName, typename... Pkgs> artdaq::BuildInfo< instanceName, Pkgs >::BuildInfo (fhicl::ParameterSet const & p) [explicit]`

`BuildInfo` module Constructor.

Parameters

<code>p</code>	ParameterSet used to configure <code>BuildInfo</code> module
----------------	--

`BuildInfo_module` expects the following Parameters: "instance_name": Name which the `BuildInfo` information will be saved under

Definition at line 82 of file `BuildInfo_module.hh`.

7.9.3 Member Function Documentation

7.9.3.1 `template<std::string * instanceName, typename... Pkgs> void artdaq::BuildInfo< instanceName, Pkgs >::beginRun (art::Run & r) [override]`

Perform actions at the beginning of the Run.

Parameters

r	art::Run object
---	-----------------

The [BuildInfo](#) information is stored in the Run-level provenance, so this method performs most of the "work" for this module.

Definition at line 92 of file [BuildInfo_module.hh](#).

7.9.3.2 template<std::string * instanceName, typename... Pkgs> void artdaq::BuildInfo< instanceName, Pkgs >::produce (art::Event & e) [override]

Perform actions for each event.

Parameters

e	art::Event object
---	-------------------

This function is a required override for EDProducer, and is a No-Op in [BuildInfo_module](#).

Definition at line 112 of file [BuildInfo_module.hh](#).

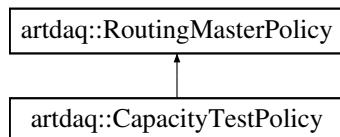
The documentation for this class was generated from the following file:

- [artdaq/artdaq/ArtModules/BuildInfo_module.hh](#)

7.10 artdaq::CapacityTestPolicy Class Reference

A [RoutingMasterPolicy](#) which tries to fully load the first receiver, then the second, and so on.

Inheritance diagram for artdaq::CapacityTestPolicy:



Public Member Functions

- [CapacityTestPolicy \(fhicl::ParameterSet ps\)](#)
CapacityTestPolicy Constructor.
- virtual [~CapacityTestPolicy \(\)=default](#)
Default virtual Destructor.
- [detail::RoutingPacket GetCurrentTable \(\) override](#)
Apply the policy to the current tokens.

Additional Inherited Members

7.10.1 Detailed Description

A [RoutingMasterPolicy](#) which tries to fully load the first receiver, then the second, and so on.

Definition at line 11 of file [CapacityTest_policy.cc](#).

7.10.2 Constructor & Destructor Documentation

7.10.2.1 artdaq::CapacityTestPolicy::CapacityTestPolicy (`fhicl::ParameterSet ps`) [explicit]

[CapacityTestPolicy](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure the CapacityTestPolicy
-----------------	---

* CapacityTestPolicy accepts the following Parameters:
 * "tokens_used_per_table_percent" (Default: 50): Percentage of available tokens to be used on each iteration.
 *

Definition at line 45 of file CapacityTest_policy.cc.

7.10.3 Member Function Documentation

7.10.3.1 `detail::RoutingPacket artdaq::CapacityTestPolicy::GetCurrentTable()` [override], [virtual]

Apply the policy to the current tokens.

Returns

A [detail::RoutingPacket](#) containing the Routing Table

[CapacityTestPolicy](#) will assign available tokens from the first receiver, then the second, and so on until it has assigned tokens equal to the `initial_token_count * tokens_used_per_table_percent / 100`. The idea is that in steady-state, the load on the receivers should reflect the workload relative to the capacity of the system. (i.e. if you have 5 receivers, and 3 of them are 100% busy, then your load factor is approximately 60%).

Implements [ardaq::RoutingMasterPolicy](#).

Definition at line 50 of file CapacityTest_policy.cc.

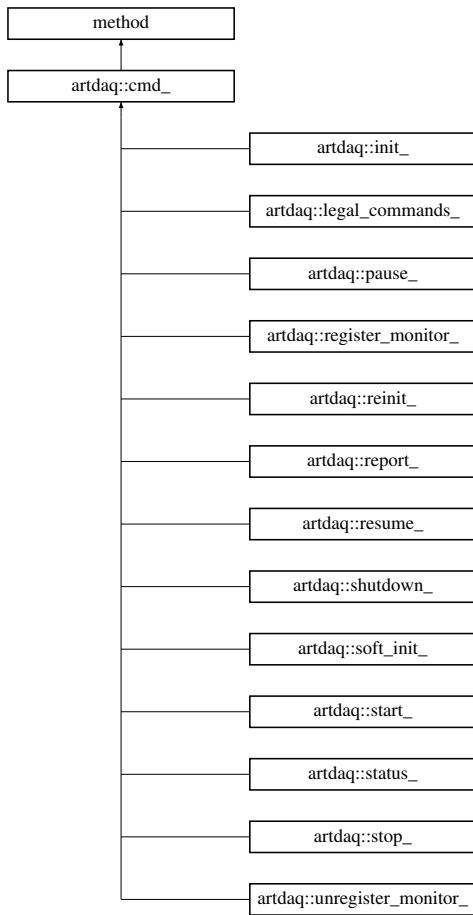
The documentation for this class was generated from the following file:

- artdaq/artdaq/Application/Routing/CapacityTest_policy.cc

7.11 artdaq::cmd_ Class Reference

The "cmd_" class serves as the base class for all artdaq's XML-RPC commands.

Inheritance diagram for artdaq::cmd_:



Public Member Functions

- `cmd_ (xmlrpc_commander &c, const std::string &signature, const std::string &description)`
`cmd_ Constructor`
- void `execute (const xmlrpc_c::paramList ¶mList, xmlrpc_c::value *const retvalP) final`
Execute trhe command with the given parameters.

Protected Member Functions

- virtual bool `execute_ (const xmlrpc_c::paramList &, xmlrpc_c::value *const retvalP)=0`
"execute_" is a wrapper function around the call to the commandable object's function
- template<typename T >
`T getParam (const xmlrpc_c::paramList ¶mList, int index)`
Get a parameter from the parameter list.
- template<typename T >
`T getParam (const xmlrpc_c::paramList ¶mList, int index, T default_value)`
Get a parameter from the parameter list, returning a default value if not found at specified location.
- template<>
`uint64_t getParam (const xmlrpc_c::paramList ¶mList, int index)`
Get a parameter from the parameter list.

Protected Attributes

- `xmlrpc_commander & _c`

The `xmlrpc_commander` instance that the command will be sent to.

7.11.1 Detailed Description

The "cmd_" class serves as the base class for all artdaq's XML-RPC commands.

JCF, 9/5/14

The "cmd_" class serves as the base class for all artdaq's XML-RPC commands, all of which use the code in the "execute()" function; each specific command type deriving from `cmd_` is implemented in the `execute_()` function which `execute()` calls (notice the underscore), and optionally sets the `retvalP` parameter

`cmd_` contains a set of template functions, `getParam<T>()`, which are designed to prevent implementors of derived classes from having to worry about interfacing directly with `xmlrpc_`'s parameter-getting functionality

Definition at line 120 of file `xmlrpc_commander.cc`.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 `artdaq::cmd_::cmd_(xmlrpc_commander & c, const std::string & signature, const std::string & description) [inline]`

`cmd_` Constructor

Parameters

<code>c</code>	<code>xmlrpc_commander</code> instance
<code>signature</code>	Signature of the command
<code>description</code>	Description of the command

Definition at line 131 of file `xmlrpc_commander.cc`.

7.11.3 Member Function Documentation

7.11.3.1 `void artdaq::cmd_::execute(const xmlrpc_c::paramList & paramList, xmlrpc_c::value *const retvalP) [final]`

Execute the command with the given parameters.

Parameters

<code>paramList</code>	List of parameters for the command (i.e. a fhicl string for init transitions)
<code>retvalP</code>	Pointer to the return value (usually a string describing result of command)

Definition at line 286 of file `xmlrpc_commander.cc`.

7.11.3.2 `virtual bool artdaq::cmd_::execute_(const xmlrpc_c::paramList &, xmlrpc_c::value *const retvalP) [protected], [pure virtual]`

"execute_" is a wrapper function around the call to the commandable object's function

Parameters

<i>retvalP</i>	Pointer to the return value (usually a string describing result of command)
----------------	---

Returns

Whether the command succeeded

7.11.3.3 `template<typename T > T artdaq::cmd_::getParam (const xmlrpc_c::paramList & paramList, int index)`
 [protected]

Get a parameter from the parameter list.

Template Parameters

<i>T</i>	Type of the parameter
----------	-----------------------

Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

Returns

The requested parameter

Template specialization is used to provide valid overloads

Definition at line 198 of file `xmlrpc_commander.cc`.

7.11.3.4 `template<typename T > T artdaq::cmd_::getParam (const xmlrpc_c::paramList & paramList, int index, T default_value)`
 [protected]

Get a parameter from the parameter list, returning a default value if not found at specified location.

Template Parameters

<i>T</i>	Type of the parameter
----------	-----------------------

Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list
<i>default_value</i>	Default value to return if exception retrieving parameter

Returns

The requested parameter, or the default value if there was an exception retrieving the parameter

JCF, 9/5/14

Here, if `getParam` throws an exception due to a lack of an existing parameter, swallow the exception and return the default value passed to the function

Surprisingly, if an invalid index is supplied, although `getParam` throws an exception that exception is neither `xmlrpc_c`'s `girerr:error` nor `boost::bad_lexical_cast`. Although it's less than ideal, we'll swallow almost all exceptions in the call to

getParam, as an invalid index value simply means the user wishes to employ the default_value. I say "almost" because the only exception we don't swallow here is if an invalid parameter type "T" was supplied

Definition at line 268 of file xmlrpc_commander.cc.

7.11.3.5 template<> uint64_t artdaq::cmd_::getParam (const xmlrpc_c::paramList & paramList, int index) [protected]

Get a parameter from the parameter list.

Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

Returns

The requested parameter

This specialized cmd_getParam for the uint64_t type

Definition at line 212 of file xmlrpc_commander.cc.

The documentation for this class was generated from the following file:

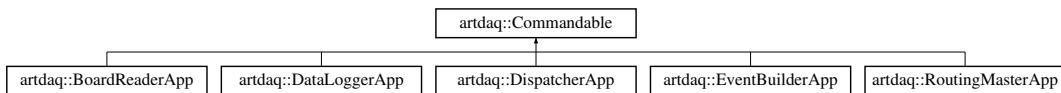
- artdaq/artdaq/ExternalComms/xmlrpc_commander.cc

7.12 artdaq::Commandable Class Reference

Commandable is the base class for all artdaq components which implement the artdaq state machine.

```
#include <artdaq/Application/Commandable.hh>
```

Inheritance diagram for artdaq::Commandable:



Public Member Functions

- **Commandable ()**
- **Commandable (Commandable const &)=delete**

Copy Constructor is deleted.
- **virtual ~Commandable ()=default**

Default Destructor.
- **Commandable & operator= (Commandable const &)=delete**

Copy Assignment operator is deleted.
- **bool initialize (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp)**

Processes the initialize request.
- **bool start (art::RunID id, uint64_t timeout, uint64_t timestamp)**

Processes the start transition.

- bool **stop** (uint64_t timeout, uint64_t timestamp)
Processes the stop transition.
- bool **pause** (uint64_t timeout, uint64_t timestamp)
Processes the pause transition.
- bool **resume** (uint64_t timeout, uint64_t timestamp)
Processes the resume transition.
- bool **shutdown** (uint64_t timeout)
Processes the shutdown transition.
- bool **soft_initialize** (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp)
Processes the soft-initialize request.
- bool **reinitialize** (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp)
Processes the reinitialize request.
- bool **in_run_failure** ()
Actions taken when the in_run_failure state is set.
- virtual std::string **report** (std::string const &) const
Default report implementation returns current report_string.
- std::string **status** () const
Returns the current state of the Commandable.
- virtual std::string **register_monitor** (fhicl::ParameterSet const &)
Perform the register_monitor action.
- virtual std::string **unregister_monitor** (std::string const &)
Perform the unregister_monitor action.
- std::vector< std::string > **legal_commands** () const
Get the legal transition commands from the current state.
- virtual bool **do_initialize** (fhicl::ParameterSet const &, uint64_t, uint64_t)
Perform the initialize transition.
- virtual bool **do_start** (art::RunID, uint64_t, uint64_t)
Perform the start transition.
- virtual bool **do_stop** (uint64_t, uint64_t)
Perform the stop transition.
- virtual bool **do_pause** (uint64_t, uint64_t)
Perform the pause transition.
- virtual bool **do_resume** (uint64_t, uint64_t)
Perform the resume transition.
- virtual bool **do_shutdown** (uint64_t)
Perform the shutdown transition.
- virtual bool **do_reinitialize** (fhicl::ParameterSet const &, uint64_t, uint64_t)
Perform the reinitialize transition.
- virtual bool **do_soft_initialize** (fhicl::ParameterSet const &, uint64_t, uint64_t)
Perform the soft_initialize transition.
- virtual void **badTransition** (const std::string &trans)
This function is called when an attempt is made to call an illegal transition.
- virtual void **BootedEnter** ()
Perform actions upon entering the Booted state.
- virtual void **InRunExit** ()
Perform actions upon leaving the InRun state.

Protected Member Functions

- std::string [current_state \(\) const](#)

Return the name of the current state.

Protected Attributes

- CommandableContext [fsm_](#)

The generated State Machine (using smc_compiler)

- bool [external_request_status_](#)

Whether the last command succeeded.

- std::string [report_string_](#)

Status information about the last command.

7.12.1 Detailed Description

[Commandable](#) is the base class for all artdaq components which implement the artdaq state machine.

Definition at line 20 of file Commandable.hh.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 artdaq::Commandable::Commandable()

Default constructor.

Definition at line 4 of file Commandable.cc.

7.12.3 Member Function Documentation

7.12.3.1 void artdaq::Commandable::badTransition(const std::string & *trans*) [virtual]

This function is called when an attempt is made to call an illegal transition.

Parameters

<i>trans</i>	The transition that was attempted
--------------	-----------------------------------

Definition at line 282 of file Commandable.cc.

7.12.3.2 void artdaq::Commandable::BootedEnter() [virtual]

Perform actions upon entering the Booted state.

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingMasterApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 301 of file Commandable.cc.

7.12.3.3 `std::string artdaq::Commandable::current_state() const` [protected]

Return the name of the current state.

Returns

The name of the current state

Definition at line 315 of file Commandable.cc.

7.12.3.4 `bool artdaq::Commandable::do_initialize(fhicl::ParameterSet const &, uint64_t, uint64_t)` [virtual]

Perform the initialize transition.

Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingMasterApp](#), [artdaq::DataLoggerApp](#), [artdaq::Dispatcher-App](#), and [artdaq::EventBuilderApp](#).

Definition at line 226 of file Commandable.cc.

7.12.3.5 `bool artdaq::Commandable::do_pause(uint64_t, uint64_t)` [virtual]

Perform the pause transition.

Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingMasterApp](#), [artdaq::DataLoggerApp](#), [artdaq::Dispatcher-App](#), and [artdaq::EventBuilderApp](#).

Definition at line 247 of file Commandable.cc.

7.12.3.6 `bool artdaq::Commandable::do_reinitialize(fhicl::ParameterSet const &, uint64_t, uint64_t)` [virtual]

Perform the reinitialize transition.

Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingMasterApp](#), [artdaq::EventBuilderApp](#), [artdaq::DataLogger-App](#), and [artdaq::DispatcherApp](#).

Definition at line 268 of file Commandable.cc.

7.12.3.7 bool artdaq::Commandable::do_resume(uint64_t , uint64_t) [virtual]

Perform the resume transition.

Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingMasterApp](#), [artdaq::DataLoggerApp](#), [artdaq::DispatcherApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 254 of file Commandable.cc.

7.12.3.8 bool artdaq::Commandable::do_shutdown(uint64_t) [virtual]

Perform the shutdown transition.

Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingMasterApp](#), [artdaq::DataLoggerApp](#), [artdaq::DispatcherApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 261 of file Commandable.cc.

7.12.3.9 bool artdaq::Commandable::do_soft_initialize(fhicl::ParameterSet const & , uint64_t , uint64_t) [virtual]

Perform the soft_initialize transition.

Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingMasterApp](#), [artdaq::EventBuilderApp](#), [artdaq::DataLoggerApp](#), and [artdaq::DispatcherApp](#).

Definition at line 275 of file Commandable.cc.

7.12.3.10 bool artdaq::Commandable::do_start(art::RunID , uint64_t , uint64_t) [virtual]

Perform the start transition.

Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingMasterApp](#), [artdaq::DataLoggerApp](#), [artdaq::DispatcherApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 233 of file Commandable.cc.

7.12.3.11 `bool artdaq::Commandable::do_stop(uint64_t , uint64_t) [virtual]`

Perform the stop transition.

Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingMasterApp](#), [artdaq::DataLoggerApp](#), [artdaq::Dispatcher-App](#), and [artdaq::EventBuilderApp](#).

Definition at line 240 of file Commandable.cc.

7.12.3.12 `bool artdaq::Commandable::in_run_failure()`

Actions taken when the in_run_failure state is set.

Returns

Whether the notification was properly processed

Definition at line 163 of file Commandable.cc.

7.12.3.13 `bool artdaq::Commandable::initialize(fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp)`

Processes the initialize request.

Parameters

<i>pset</i>	ParameterSet used to configure the Commandable
<i>timeout</i>	Timeout for init step
<i>timestamp</i>	Timestamp of init step

Returns

Whether the transition was successful

Definition at line 11 of file Commandable.cc.

7.12.3.14 `void artdaq::Commandable::InRunExit() [virtual]`

Perform actions upon leaving the InRun state.

This function is a No-Op. Derived classes should override it.

Definition at line 306 of file Commandable.cc.

7.12.3.15 `std::vector< std::string > artdaq::Commandable::legal_commands() const`

Get the legal transition commands from the current state.

Returns

A list of legal transitions from the current state

Definition at line 194 of file Commandable.cc.

7.12.3.16 Commandable& artdaq::Commandable::operator= (Commandable const &) [delete]

Copy Assignment operator is deleted.

Returns

[Commandable](#) copy

7.12.3.17 bool artdaq::Commandable::pause (uint64_t timeout, uint64_t timestamp)

Processes the pause transition.

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition was successful

Definition at line 68 of file Commandable.cc.

7.12.3.18 virtual std::string artdaq::Commandable::register_monitor (fhicl::ParameterSet const &) [inline], [virtual]

Perform the register_monitor action.

Returns

A report on the status of the command

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::DispatcherApp](#).

Definition at line 139 of file Commandable.hh.

7.12.3.19 bool artdaq::Commandable::reinitialize (fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp)

Processes the reinitialize request.

Parameters

<i>pset</i>	ParameterSet used to configure the Commandable
-------------	--

<i>timeout</i>	Timeout for init step
<i>timestamp</i>	Timestamp of init step

Returns

Whether the transition was successful

Definition at line 144 of file Commandable.cc.

7.12.3.20 virtual std::string artdaq::Commandable::report(std::string const &) const [inline], [virtual]

Default report implementation returns current report_string.

Returns

Current report_string (may be set by derived classes)

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingMasterApp](#), [artdaq::EventBuilderApp](#), [artdaq::DataLogger-App](#), and [artdaq::DispatcherApp](#).

Definition at line 121 of file Commandable.hh.

7.12.3.21 bool artdaq::Commandable::resume(uint64_t timeout, uint64_t timestamp)

Processes the resume transition.

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition was successful

Definition at line 87 of file Commandable.cc.

7.12.3.22 bool artdaq::Commandable::shutdown(uint64_t timeout)

Processes the shutdown transition.

Parameters

<i>timeout</i>	Timeout for transition
----------------	------------------------

Returns

Whether the transition was successful

Definition at line 106 of file Commandable.cc.

7.12.3.23 bool artdaq::Commandable::soft_initialize(fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp)

Processes the soft-initialize request.

Parameters

<i>pset</i>	ParameterSet used to configure the Commandable
<i>timeout</i>	Timeout for init step
<i>timestamp</i>	Timestamp of init step

Returns

Whether the transition was successful

Definition at line 125 of file Commandable.cc.

7.12.3.24 `bool artdaq::Commandable::start (art::RunID id, uint64_t timeout, uint64_t timestamp)`

Processes the start transition.

Parameters

<i>id</i>	Run number of new run
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition was successful

Definition at line 30 of file Commandable.cc.

7.12.3.25 `std::string artdaq::Commandable::status () const`

Returns the current state of the [Commandable](#).

Returns

The current state of the [Commandable](#)

Definition at line 182 of file Commandable.cc.

7.12.3.26 `bool artdaq::Commandable::stop (uint64_t timeout, uint64_t timestamp)`

Processes the stop transition.

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition was successful

Definition at line 49 of file Commandable.cc.

7.12.3.27 virtual std::string artdaq::Commandable::unregister_monitor(std::string const &) [inline], [virtual]

Perform the unregister_monitor action.

Returns

A report on the status of the command

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::DispatcherApp](#).

Definition at line 150 of file Commandable.hh.

The documentation for this class was generated from the following files:

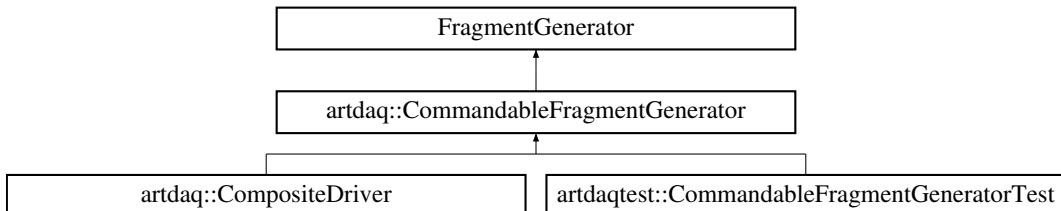
- artdaq/artdaq/Application/Commandable.hh
- artdaq/artdaq/Application/Commandable.cc

7.13 artdaq::CommandableFragmentGenerator Class Reference

[CommandableFragmentGenerator](#) is a FragmentGenerator-derived abstract class that defines the interface for a FragmentGenerator designed as a state machine with start, stop, etc., transition commands.

```
#include <artdaq/Application/CommandableFragmentGenerator.hh>
```

Inheritance diagram for artdaq::CommandableFragmentGenerator:



Public Member Functions

- [CommandableFragmentGenerator\(\)](#)
CommandableFragmentGenerator default constructor.
- [CommandableFragmentGenerator\(const fhicl::ParameterSet &ps\)](#)
CommandableFragmentGenerator Constructor.
- [virtual ~CommandableFragmentGenerator\(\)](#)
CommandableFragmentGenerator Destructor.
- [bool getNext\(FragmentPtrs &output\) overridefinal](#)
getNext calls either applyRequests or getNext_ to get any data that is ready to be sent to the EventBuilders
- [void applyRequestsIgnoredMode\(artdaq::FragmentPtrs &frags\)](#)
Create fragments using data buffer for request mode Ignored. Precondition: dataBufferMutex_ and request_mutex_ are locked
- [void applyRequestsSingleMode\(artdaq::FragmentPtrs &frags\)](#)
Create fragments using data buffer for request mode Single. Precondition: dataBufferMutex_ and request_mutex_ are locked

- void [applyRequestsBufferMode](#) (artdaq::FragmentPtrs &frags)

Create fragments using data buffer for request mode Buffer. Precondition: dataBufferMutex_ and request_mutex_ are locked
- void [applyRequestsWindowMode](#) (artdaq::FragmentPtrs &frags)

Create fragments using data buffer for request mode Window. Precondition: dataBufferMutex_ and request_mutex_ are locked
- bool [applyRequests](#) (FragmentPtrs &output)

See if any requests have been received, and add the corresponding data Fragment objects to the output list.
- void [setupRequestListener](#) ()

Opens the socket used to listen for data requests.
- bool [sendEmptyFragment](#) (FragmentPtrs &frags, size_t sequenceld, std::string desc)

Send an EmptyFragmentType Fragment.
- void [sendEmptyFragments](#) (FragmentPtrs &frags)

This function is for Buffered and Single request modes, as they can only respond to one data request at a time If the request message seqID > ev_counter, simply send empties until they're equal.
- void [startDataThread](#) ()

Function that launches the data thread ([getDataLoop\(\)](#))
- void [startMonitoringThread](#) ()

Function that launches the monitoring thread ([getMonitoringDataLoop\(\)](#))
- void [startRequestReceiverThread](#) ()

Function that launches the data request receiver thread ([receiveRequestsLoop\(\)](#))
- void [getDataLoop](#) ()

When separate_data_thread is set to true, this loop repeatedly calls [getNext_](#) and adds returned Fragment objects to the data buffer, blocking when the data buffer is full.
- bool [waitForDataBufferReady](#) ()

Wait for the data buffer to drain ([dataBufferIsTooLarge](#) returns false), periodically reporting status.
- bool [dataBufferIsTooLarge](#) ()

Test the configured constraints on the data buffer.
- void [getDataBufferStats](#) ()

Calculate the size of the dataBuffer and report appropriate metrics.
- void [checkDataBuffer](#) ()

Perform data buffer pruning operations. If the RequestMode is Single, removes all but the latest Fragment from the data buffer. In Window and Buffer RequestModes, this function discards the oldest Fragment objects until the data buffer is below its size constraints, then also checks for stale Fragments, based on the timestamp of the most recent Fragment.
- void [getMonitoringDataLoop](#) ()

This function regularly calls [checkHWStatus_\(\)](#), and sets the [isHardwareOK](#) flag accordingly.
- void [receiveRequestsLoop](#) ()

This function receives data request packets, adding new requests to the request list.
- std::vector<Fragment::fragment_id_t> [fragmentIDs](#) () override

Get the list of Fragment IDs handled by this [CommandableFragmentGenerator](#).
- void [StartCmd](#) (int run, uint64_t timeout, uint64_t timestamp)

Start the [CommandableFragmentGenerator](#).
- void [StopCmd](#) (uint64_t timeout, uint64_t timestamp)

Stop the [CommandableFragmentGenerator](#).
- void [PauseCmd](#) (uint64_t timeout, uint64_t timestamp)

Pause the [CommandableFragmentGenerator](#).
- void [ResumeCmd](#) (uint64_t timeout, uint64_t timestamp)

Resume the CommandableFragmentGenerator.

- std::string [ReportCmd](#) (std::string const &which="")

Get a report about a user-specified run-time quantity.
- virtual std::string [metricsReportingInstanceName](#) () const

Get the name used when reporting metrics.
- bool [exception](#) () const

Get the current value of the exception flag.

Protected Member Functions

- int [run_number](#) () const

Get the current Run number.
- int [subrun_number](#) () const

Get the current Subrun number.
- uint64_t [timeout](#) () const

Timeout of last command.
- uint64_t [timestamp](#) () const

Timestamp of last command.
- bool [should_stop](#) () const

Get the current value of the should_stop flag.
- bool [check_stop](#) ()

Routine used by applyRequests to make sure that all outstanding requests have been fulfilled before returning.
- int [board_id](#) () const

Gets the current board_id.
- int [fragment_id](#) () const

Get the current Fragment ID, if there is only one.
- size_t [ev_counter](#) () const

Get the current value of the event counter.
- size_t [ev_counter_inc](#) (size_t step=1, bool force=false)

Increment the event counter, if the current RequestMode allows it.
- void [set_exception](#) (bool [exception](#))

Control the exception flag.
- void [metricsReportingInstanceName](#) (std::string const &name)

Sets the name for metrics reporting.
- std::string [printMode_](#) ()

Return the string representation of the current RequestMode.

Protected Attributes

- std::mutex [mutex_](#)

Mutex used to ensure that multiple transition commands do not run at the same time.

7.13.1 Detailed Description

[CommandableFragmentGenerator](#) is a FragmentGenerator-derived abstract class that defines the interface for a FragmentGenerator designed as a state machine with start, stop, etc., transition commands.

Users of classes derived from [CommandableFragmentGenerator](#) will call these transitions via the publically defined [StartCmd\(\)](#), [StopCmd\(\)](#), etc.; these public functions contain functionality considered properly universal to all CommandableFragmentGenerator-derived classes, including calls to private virtual functions meant to be overridden in derived classes. The same applies to this class's implementation of the FragmentGenerator::getNext() pure virtual function, which is declared final (i.e., non-overridable in derived classes) and which itself calls a pure virtual getNext_() function to be implemented in derived classes.

State-machine related interface functions will be called only from a single thread. [getNext\(\)](#) will be called only from a single thread. The thread from which state-machine interfaces functions are called may be a different thread from the one that calls [getNext\(\)](#).

John F., 3/24/14

After some discussion with Kurt, [CommandableFragmentGenerator](#) has been updated such that it now contains a member vector fragment_ids_; if "fragment_id" is set in the FHiCL document controlling a class derived from [CommandableFragmentGenerator](#), fragment_ids_ will be booked as a length-1 vector, and the value in this vector will be returned by [fragment_id\(\)](#). [fragment_id\(\)](#) will throw an exception if the length of the vector isn't 1. If "fragment_ids" is set in the FHiCL document, then fragment_ids_ is filled with the values in the list which "fragment_ids" refers to, otherwise it is set to the empty vector (this is what should happen if the user sets the "fragment_id" variable in the FHiCL document, otherwise exceptions will end up thrown due to the logical conflict). If neither "fragment_id" nor "fragment_ids" is set in the FHiCL document, writers of classes derived from this one will be expected to override the virtual [fragmentIDs\(\)](#) function with their own code (the [CompositeDriver](#) class is an example of this)

Definition at line 83 of file CommandableFragmentGenerator.hh.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 artdaq::CommandableFragmentGenerator::CommandableFragmentGenerator()

[CommandableFragmentGenerator](#) default constructor.

This constructor defalt-initializes all parameters

Definition at line 26 of file CommandableFragmentGenerator.cc.

7.13.2.2 artdaq::CommandableFragmentGenerator::CommandableFragmentGenerator(const fhi::ParameterSet & ps) [explicit]

[CommandableFragmentGenerator](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure CommandableFragmentGenerator
-----------------	---

- * CommandableFragmentGenerator accepts the following Parameters:
- * "request_port" (Default: 3001): Port on which data requests will be received
- * "request_address" (Default: "227.128.12.26"): Address which CommandableFragmentGenerator will listen for requests
- * "end_of_run_quiet_timeout_ms" (Default: 1000): Time, in milliseconds, that the entire system must be quiet for
- * "request_window_offset" (Default: 0): Request messages contain a timestamp. For Window request mode, start the window at this offset
- * "request_window_width" (Default: 0): For Window request mode, the window will be timestamp - offset to timestamp
- * "stale_request_timeout" (Default: -1): How long should request messages be retained
- * "request_windows_are_unique" (Default: true): Whether Fragments should be removed from the buffer when matched
- * "missing_request_window_timeout_us" (Default: 1s): How long to wait for a missing request in Window mode (measured in microseconds)
- * "window_close_timeout_us" (Default: 2s): How long to wait for the end of the data buffer to pass the end of a request

```
* "expected_fragment_type" (Default: 231, EmptyFragmentType): The type of Fragments this CFG will be generating.
* "separate_data_thread" (Default: false): Whether data collection should proceed on its own thread. Required for
* "sleep_on_no_data_us" (Default: 0 (no sleep)): How long to sleep after calling getNext_ if no data is returned
* "data_buffer_depth_fragments" (Default: 1000): How many Fragments to store in the buffer
* "data_buffer_depth_mb" (Default: 1000): The maximum size of the data buffer in MB
* "separate_monitoring_thread" (Default: false): Whether a thread that calls the checkHWStatus_ method should be created
* "hardware_poll_interval_us" (Default: 0): If a separate monitoring thread is used, how often should it call checkHWStatus_
* "board_id" (REQUIRED): The identification number for this CommandableFragmentGenerator
* "fragment_ids" (Default: empty vector): A list of Fragment IDs created by this CommandableFragmentGenerator
* "fragment_id" (Default: -99): The Fragment ID created by this CommandableFragmentGenerator
* Note that only one of fragment_ids and fragment_id should be specified in the configuration
* "sleep_on_stop_us" (Default: 0): How long to sleep before returning when stop transition is called
* "request_mode" (Default: Ignored): The mode by which the CommandableFragmentGenerator will process requests
* Ignored: Request messages are ignored. This is a "push" CommandableFragmentGenerator
* Single: The CommandableFragmentGenerator responds to each request with the latest Fragment it has received
* Buffer: The CommandableFragmentGenerator responds to each request with all Fragments it has received since the last request
* Window: The CommandableFragmentGenerator searches its data buffer for all Fragments whose timestamp falls within the requested window
```

Definition at line 73 of file CommandableFragmentGenerator.cc.

7.13.2.3 artdaq::CommandableFragmentGenerator::~CommandableFragmentGenerator() [virtual]

CommandableFragmentGenerator Destructor.

Joins all threads before returning

Definition at line 221 of file CommandableFragmentGenerator.cc.

7.13.3 Member Function Documentation

7.13.3.1 bool artdaq::CommandableFragmentGenerator::applyRequests(FragmentPtrs & output)

See if any requests have been received, and add the corresponding data Fragment objects to the output list.

Parameters

out	output	list of FragmentPtr objects ready for transmission
-----	--------	--

Returns

True if not stopped

Definition at line 1000 of file CommandableFragmentGenerator.cc.

7.13.3.2 void artdaq::CommandableFragmentGenerator::applyRequestsBufferMode(artdaq::FragmentPtrs & frags)

Create fragments using data buffer for request mode Buffer. Precondition: dataBufferMutex_ and request_mutex_ are locked

Parameters

frags	Ouput fragments
-------	-----------------

Definition at line 873 of file CommandableFragmentGenerator.cc.

7.13.3.3 void artdaq::CommandableFragmentGenerator::applyRequestsIgnoredMode (artdaq::FragmentPtrs & frags)

Create fragments using data buffer for request mode Ignored. Precondition: dataBufferMutex_ and request_mutex_ are locked

Parameters

<i>frags</i>	Ouput fragments
--------------	-----------------

Definition at line 834 of file CommandableFragmentGenerator.cc.

7.13.3.4 void artdaq::CommandableFragmentGenerator::applyRequestsSingleMode (artdaq::FragmentPtrs & *frags*)

Create fragments using data buffer for request mode Single. Precondition: dataBufferMutex_ and request_mutex_ are locked

Parameters

<i>frags</i>	Ouput fragments
--------------	-----------------

Definition at line 842 of file CommandableFragmentGenerator.cc.

7.13.3.5 void artdaq::CommandableFragmentGenerator::applyRequestsWindowMode (artdaq::FragmentPtrs & *frags*)

Create fragments using data buffer for request mode Window. Precondition: dataBufferMutex_ and request_mutex_ are locked

Parameters

<i>frags</i>	Ouput fragments
--------------	-----------------

Definition at line 898 of file CommandableFragmentGenerator.cc.

7.13.3.6 int artdaq::CommandableFragmentGenerator::board_id () const [inline], [protected]

Gets the current board_id.

Returns

The current board_id

Definition at line 407 of file CommandableFragmentGenerator.hh.

7.13.3.7 bool artdaq::CommandableFragmentGenerator::check_stop () [protected]

Routine used by applyRequests to make sure that all outstanding requests have been fulfilled before returning.

Returns

The logical AND of should_stop, mode is not Ignored, and requests list size equal to 0

Definition at line 328 of file CommandableFragmentGenerator.cc.

7.13.3.8 bool artdaq::CommandableFragmentGenerator::dataBufferIsTooLarge ()

Test the configured constraints on the data buffer.

Returns

Whether the data buffer is full

Definition at line 661 of file CommandableFragmentGenerator.cc.

7.13.3.9 size_t artdaq::CommandableFragmentGenerator::ev_counter() const [inline], [protected]

Get the current value of the event counter.

Returns

The current value of the event counter

Definition at line 420 of file CommandableFragmentGenerator.hh.

7.13.3.10 size_t artdaq::CommandableFragmentGenerator::ev_counter_inc(size_t step = 1, bool force = false) [protected]

Increment the event counter, if the current RequestMode allows it.

Parameters

<i>step</i>	Amount to increment the event counter by
<i>force</i>	Force incrementing the event Counter

Returns

The previous value of the event counter

Definition at line 360 of file CommandableFragmentGenerator.cc.

7.13.3.11 bool artdaq::CommandableFragmentGenerator::exception() const [inline]

Get the current value of the exception flag.

Returns

The current value of the exception flag

Definition at line 358 of file CommandableFragmentGenerator.hh.

7.13.3.12 int artdaq::CommandableFragmentGenerator::fragment_id() const [protected]

Get the current Fragment ID, if there is only one.

Returns

The current fragment ID, if there is only one

Exceptions

<i>cet::exception</i>	if the Fragment IDs list has more than one member
-----------------------	---

Definition at line 348 of file CommandableFragmentGenerator.cc.

7.13.3.13 std::vector<Fragment::fragment_id_t> artdaq::CommandableFragmentGenerator::fragmentIDs() [inline], [override]

Get the list of Fragment IDs handled by this [CommandableFragmentGenerator](#).

Returns

A std::vector<Fragment::fragment_id_t> containing the Fragment IDs handled by this [CommandableFragmentGenerator](#)

Definition at line 260 of file CommandableFragmentGenerator.hh.

7.13.3.14 void artdaq::CommandableFragmentGenerator::getDataBufferStats()

Calculate the size of the dataBuffer and report appropriate metrics.

dataBufferMutex must be owned by the calling thread!

Definition at line 666 of file CommandableFragmentGenerator.cc.

7.13.3.15 bool artdaq::CommandableFragmentGenerator::getNext(FragmentPtrs & output) [final], [override]

getNext calls either applyRequests or getNext_ to get any data that is ready to be sent to the EventBuilders

Parameters

<i>output</i>	FragmentPtrs object containing Fragments ready for transmission
---------------	---

Returns

Whether getNext completed without exceptions

Definition at line 234 of file CommandableFragmentGenerator.cc.

7.13.3.16 virtual std::string artdaq::CommandableFragmentGenerator::metricsReportingInstanceName() const [inline], [virtual]

Get the name used when reporting metrics.

Returns

The name used when reporting metrics

Definition at line 333 of file CommandableFragmentGenerator.hh.

7.13.3.17 void artdaq::CommandableFragmentGenerator::metricsReportingInstanceName(std::string const & name) [inline], [protected]

Sets the name for metrics reporting.

Parameters

<i>name</i>	The new name for metrics reporting
-------------	------------------------------------

Definition at line 440 of file CommandableFragmentGenerator.hh.

7.13.3.18 void artdaq::CommandableFragmentGenerator::PauseCmd(uint64_t timeout, uint64_t timestamp)

Pause the [CommandableFragmentGenerator](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

A call to [PauseCmd\(\)](#) is advisory. It is an indication that the BoardReader should stop the incoming flow of data, if it can do so.

Definition at line 406 of file CommandableFragmentGenerator.cc.

7.13.3.19 std::string artdaq::CommandableFragmentGenerator::printMode_ () [protected]

Return the string representation of the current RequestMode.

Returns

The string representation of the current RequestMode

Definition at line 523 of file CommandableFragmentGenerator.cc.

7.13.3.20 std::string artdaq::CommandableFragmentGenerator::ReportCmd (std::string const & which = " ")

Get a report about a user-specified run-time quantity.

Parameters

<i>which</i>	Which quantity to report
--------------	--------------------------

Returns

The report about the specified quantity

[CommandableFragmentGenerator](#) only implements "latest_exception", a report on the last exception received. However, child classes can override the reportSpecific function to provide additional reports.

Definition at line 438 of file CommandableFragmentGenerator.cc.

7.13.3.21 void artdaq::CommandableFragmentGenerator::ResumeCmd (uint64_t timeout, uint64_t timestamp)

Resume the [CommandableFragmentGenerator](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

After a call to [ResumeCmd\(\)](#), the next Fragments returned from [getNext\(\)](#) will be part of a new SubRun.

Definition at line 418 of file CommandableFragmentGenerator.cc.

7.13.3.22 int artdaq::CommandableFragmentGenerator::run_number () const [inline], [protected]

Get the current Run number.

Returns

The current Run number

Definition at line 374 of file CommandableFragmentGenerator.hh.

7.13.3.23 `bool artdaq::CommandableFragmentGenerator::sendEmptyFragment (FragmentPtrs & frags, size_t sequenceld, std::string desc)`

Send an EmptyFragmentType Fragment.

Parameters

<code>out</code>	<code>frags</code>	Output list to append EmptyFragmentType to
	<code>sequenceld</code>	Sequence ID of Empty Fragment
	<code>desc</code>	Message to log with reasoning for sending Empty Fragment

Returns

True if no exceptions

Definition at line 1062 of file CommandableFragmentGenerator.cc.

7.13.3.24 `void artdaq::CommandableFragmentGenerator::sendEmptyFragments (FragmentPtrs & frags)`

This function is for Buffered and Single request modes, as they can only respond to one data request at a time If the request message seqID > ev_counter, simply send empties until they're equal.

Parameters

<code>out</code>	<code>frags</code>	Output list to append EmptyFragmentType to
------------------	--------------------	--

Definition at line 1076 of file CommandableFragmentGenerator.cc.

7.13.3.25 `void artdaq::CommandableFragmentGenerator::set_exception (bool exception) [inline], [protected]`

Control the exception flag.

Parameters

<code>exception</code>	Whether an exception has occurred
------------------------	-----------------------------------

Definition at line 434 of file CommandableFragmentGenerator.hh.

7.13.3.26 `bool artdaq::CommandableFragmentGenerator::should_stop () const [inline], [protected]`

Get the current value of the should_stop flag.

Returns

The current value of the should_stop flag

Definition at line 395 of file CommandableFragmentGenerator.hh.

7.13.3.27 `void artdaq::CommandableFragmentGenerator::StartCmd (int run, uint64_t timeout, uint64_t timestamp)`

Start the [CommandableFragmentGenerator](#).

Parameters

<i>run</i>	Run ID of the new run
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

After a call to 'StartCmd', all Fragments returned by [getNext\(\)](#) will be marked as part of a Run with the given run number, and with subrun number 1. Calling StartCmd also resets the event number to 1. After a call to [StartCmd\(\)](#), and until a call to StopCmd, [getNext\(\)](#) – and hence the virtual function it calls, [getNext_\(\)](#) – should return true as long as datataking is meant to take place, even if a particular call returns no fragments.

Definition at line 369 of file CommandableFragmentGenerator.cc.

7.13.3.28 void artdaq::CommandableFragmentGenerator::StopCmd (uint64_t timeout, uint64_t timestamp)

Stop the [CommandableFragmentGenerator](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

After a call to [StopCmd\(\)](#), [getNext\(\)](#) will eventually return false. This may not happen for several calls, if the implementation has data to be 'drained' from the system.

Definition at line 393 of file CommandableFragmentGenerator.cc.

7.13.3.29 int artdaq::CommandableFragmentGenerator::subrun_number () const [inline], [protected]

Get the current Subrun number.

Returns

The current Subrun number

Definition at line 379 of file CommandableFragmentGenerator.hh.

7.13.3.30 uint64_t artdaq::CommandableFragmentGenerator::timeout () const [inline], [protected]

[Timeout](#) of last command.

Returns

[Timeout](#) of last command

Definition at line 384 of file CommandableFragmentGenerator.hh.

7.13.3.31 uint64_t artdaq::CommandableFragmentGenerator::timestamp () const [inline], [protected]

Timestamp of last command.

Returns

Timestamp of last command

Definition at line 389 of file CommandableFragmentGenerator.hh.

7.13.3.32 bool artdaq::CommandableFragmentGenerator::waitForDataBufferReady ()

Wait for the data buffer to drain (dataBufferIsTooLarge returns false), periodically reporting status.

Returns

True if wait ended without something else disrupting the run

Definition at line 628 of file CommandableFragmentGenerator.cc.

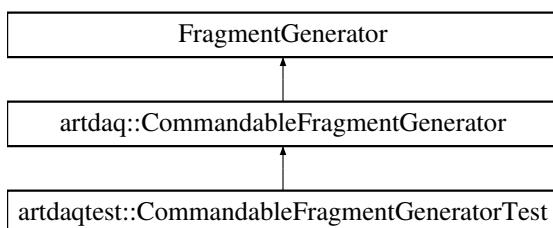
The documentation for this class was generated from the following files:

- artdaq/artdaq/Application/CommandableFragmentGenerator.hh
- artdaq/artdaq/Application/CommandableFragmentGenerator.cc

7.14 artdaqtest::CommandableFragmentGeneratorTest Class Reference

CommandableFragmentGenerator derived class for testing.

Inheritance diagram for artdaqtest::CommandableFragmentGeneratorTest:



Public Member Functions

- **CommandableFragmentGeneratorTest** (const fhicl::ParameterSet &ps)
CommandableFragmentGeneratorTest Constructor.
- bool **getNext_** (artdaq::FragmentPtrs &frags) override
Generate data and return it to CommandableFragmentGenerator.
- bool **checkHWStatus_** () override
Returns whether the hwFail flag has not been set.
- void **start** () override
Perform start actions. No-Op.
- void **stopNoMutex** () override
Perform immediate stop actions. No-Op.
- void **stop** () override
Perform stop actions. No-Op.
- void **pause** () override
Perform pause actions. No-Op.
- void **resume** () override
Perform resume actions. No-Op.
- void **setFireCount** (size_t count)
Have getNext_ generate count fragments.

- void [setHwFail \(\)](#)
Set the hwFail flag.
- void [waitForFrags \(\)](#)
Wait for all fragments generated to be read by the CommandableFragmentGenerator

Additional Inherited Members

7.14.1 Detailed Description

CommandableFragmentGenerator derived class for testing.

Definition at line 19 of file CommandableFragmentGenerator_t.cc.

7.14.2 Member Function Documentation

7.14.2.1 bool artdaqtest::CommandableFragmentGeneratorTest::checkHWStatus_ () [inline], [override], [virtual]

Returns whether the hwFail flag has not been set.

Returns

If hwFail has been set, false, otherwise true

Reimplemented from [artdaq::CommandableFragmentGenerator](#).

Definition at line 43 of file CommandableFragmentGenerator_t.cc.

7.14.2.2 bool artdaqtest::CommandableFragmentGeneratorTest::getNext_ (artdaq::FragmentPtrs & frags) [override]

Generate data and return it to CommandableFragmentGenerator.

Parameters

<i>frags</i>	FragmentPtrs list that new Fragments should be added to
--------------	---

Returns

True if data was generated

[CommandableFragmentGeneratorTest](#) merely default-constructs Fragments, emplacing them on the frags list.

Definition at line 105 of file CommandableFragmentGenerator_t.cc.

7.14.2.3 void artdaqtest::CommandableFragmentGeneratorTest::setFireCount (size_t count) [inline]

Have getNext_ generate count fragments.

Parameters

<code>count</code>	Number of fragments to generate
--------------------	---------------------------------

Definition at line 74 of file CommandableFragmentGenerator_t.cc.

7.14.2.4 void artdaqtest::CommandableFragmentGeneratorTest::waitForFrgs() [inline]

Wait for all fragments generated to be read by the CommandableFragmentGenerator

Definition at line 84 of file CommandableFragmentGenerator_t.cc.

The documentation for this class was generated from the following file:

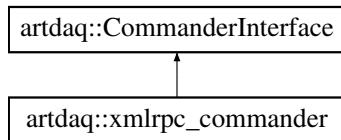
- artdaq/test/Application/CommandableFragmentGenerator_t.cc

7.15 artdaq::CommanderInterface Class Reference

This interface defines the functions used to transfer data between artdaq applications.

```
#include <artdaq/ExternalComms/CommanderInterface.hh>
```

Inheritance diagram for artdaq::CommanderInterface:



Public Member Functions

- `CommanderInterface` (const fhicl::ParameterSet &ps, `artdaq::Commandable` &commandable)
CommanderInterface Constructor.
- `CommanderInterface` (const `CommanderInterface` &)=delete
Copy Constructor is deleted.
- `CommanderInterface` & `operator=` (const `CommanderInterface` &)=delete
Copy Assignment operator is deleted.
- virtual `~CommanderInterface` ()=default
Default virtual Destructor.
- virtual void `run_server` ()=0
run_server is the main work loop for the Commander.
- virtual std::string `send_init` (fhicl::ParameterSet, uint64_t, uint64_t)
Using the transport mechanism, send an init command
- virtual std::string `send_soft_init` (fhicl::ParameterSet, uint64_t, uint64_t)
Using the transport mechanism, send a soft_init command
- virtual std::string `send_reinit` (fhicl::ParameterSet, uint64_t, uint64_t)
Using the transport mechanism, send a reinit command
- virtual std::string `send_start` (art::RunID, uint64_t, uint64_t)
Using the transport mechanism, send a start command

- virtual std::string `send_pause` (uint64_t, uint64_t)
Using the transport mechanism, send a pause command
- virtual std::string `send_resume` (uint64_t, uint64_t)
Using the transport mechanism, send a resume command
- virtual std::string `send_stop` (uint64_t, uint64_t)
Using the transport mechanism, send a stop command
- virtual std::string `send_shutdown` (uint64_t)
Using the transport mechanism, send a shutdown command
- virtual std::string `send_status` ()
Using the transport mechanism, send a status command
- virtual std::string `send_report` (std::string)
Using the transport mechanism, send a report command
- virtual std::string `send_legal_commands` ()
Using the transport mechanism, send a legal_commands command
- virtual std::string `send_register_monitor` (std::string)
Using the transport mechanism, send a register_monitor command
- virtual std::string `send_unregister_monitor` (std::string)
Using the transport mechanism, send an unregister_monitor command

Public Attributes

- `artdaq::Commandable & _commandable`
Reference to the `Commandable` that this Commander Commands.

Protected Attributes

- int `_id`
ID Number of this Commander.

7.15.1 Detailed Description

This interface defines the functions used to transfer data between artdaq applications.

Definition at line 13 of file CommanderInterface.hh.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 `artdaq::CommanderInterface::CommanderInterface (const fhicl::ParameterSet & ps, artdaq::Commandable & commandable) [inline]`

`CommanderInterface` Constructor.

Parameters

<i>ps</i>	ParameterSet used for configuring the CommanderInterface
<i>commandable</i>	artdaq::Commandable object to send transition commands to

* CommanderInterface accepts the following Parameters:
 "commanderPluginType": The type of Commander plugin to load
 "id": Integer ID number of this Commandable. May be constrained by plugin types (i.e. XMLRPC port number).
 *

Definition at line 29 of file CommanderInterface.hh.

7.15.3 Member Function Documentation

7.15.3.1 CommanderInterface& artdaq::CommanderInterface::operator=(const CommanderInterface &) [delete]

Copy Assignment operator is deleted.

Returns

[CommanderInterface](#) Copy

7.15.3.2 virtual void artdaq::CommanderInterface::run_server() [pure virtual]

run_server is the main work loop for the Commander.

This function is expected to block and persist for the entire run of the application. It should accept and handle the following commands (subject to state-machine constraints, see [Commandable::legal_commands\(\)](#)): init soft_init reinit start pause resume stop shutdown status report legal_commands register_monitor unregister_monitor

See the send_* functions for more details on each command. Not all commands are valid for all applications/states. run_server should return a string indicating success or failure to the transport mechanism when it is done processing a command.

Implemented in [artdaq::xmlrpc_commander](#).

7.15.3.3 std::string artdaq::CommanderInterface::send_init(fhicl::ParameterSet , uint64_t , uint64_t) [inline], [virtual]

Using the transport mechanism, send an init command

The init command is accepted by all artdaq processes that are in the booted state. It expects a ParameterSet for configuration, a timeout, and a timestamp.

Returns

Command result: "SUCCESS" if succeeded

Definition at line 4 of file CommanderInterface.cc.

7.15.3.4 std::string artdaq::CommanderInterface::send_legal_commands() [inline], [virtual]

Using the transport mechanism, send a legal_commands command

This will query the artdaq process, and it will return the list of allowed transition commands from its current state.

Returns

Command result: a list of allowed transition commands from its current state

Definition at line 64 of file CommanderInterface.cc.

7.15.3.5 std::string artdaq::CommanderInterface::send_pause(uint64_t , uint64_t) [inline], [virtual]

Using the transport mechanism, send a pause command

The pause command pauses a Run. When the run resumes, the subrun number will be incremented. This command accepts a timeout parameter and a timestamp parameter.

Returns

Command result: "SUCCESS" if succeeded

Definition at line 28 of file CommanderInterface.cc.

7.15.3.6 std::string artdaq::CommanderInterface::send_register_monitor(std::string) [inline], [virtual]

Using the transport mechanism, send a register_monitor command

This will cause a Dispatcher to start an art process with the given FHiCL configuration string

Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc_commander](#).

Definition at line 70 of file CommanderInterface.cc.

7.15.3.7 std::string artdaq::CommanderInterface::send_reinit(fhicl::ParameterSet , uint64_t , uint64_t) [inline], [virtual]

Using the transport mechanism, send a reinit command

The reinit command is accepted by all artdaq processes. It expects a ParameterSet for configuration, a timeout, and a timestamp.

Returns

Command result: "SUCCESS" if succeeded

Definition at line 16 of file CommanderInterface.cc.

7.15.3.8 std::string artdaq::CommanderInterface::send_report(std::string) [inline], [virtual]

Using the transport mechanism, send a report command

The report command returns the current value of the requested reportable quantity.

Returns

Command result: current value of the requested reportable quantity

Definition at line 58 of file CommanderInterface.cc.

7.15.3.9 std::string artdaq::CommanderInterface::send_resume(uint64_t , uint64_t) [inline], [virtual]

Using the transport mechanism, send a resume command

The resume command resumes a paused Run. When the run resumes, the subrun number will be incremented. This command accepts a timeout parameter and a timestamp parameter.

Returns

Command result: "SUCCESS" if succeeded

Definition at line 34 of file CommanderInterface.cc.

7.15.3.10 std::string artdaq::CommanderInterface::send_shutdown(uint64_t) [inline], [virtual]

Using the transport mechanism, send a shutdown command

The shutdown command shuts down the artdaq process. This command accepts a timeout parameter.

Returns

Command result: "SUCCESS" if succeeded

Definition at line 46 of file CommanderInterface.cc.

7.15.3.11 std::string artdaq::CommanderInterface::send_soft_init(fhicl::ParameterSet , uint64_t , uint64_t) [inline], [virtual]

Using the transport mechanism, send a soft_init command

The soft_init command is accepted by all artdaq processes that are in the booted state. It expects a ParameterSet for configuration, a timeout, and a timestamp.

Returns

Command result: "SUCCESS" if succeeded

Definition at line 10 of file CommanderInterface.cc.

7.15.3.12 std::string artdaq::CommanderInterface::send_start(art::RunID , uint64_t , uint64_t) [inline], [virtual]

Using the transport mechanism, send a start command

The start command starts a Run using the given run number. This command also accepts a timeout parameter and a timestamp parameter.

Returns

Command result: "SUCCESS" if succeeded

Definition at line 22 of file CommanderInterface.cc.

7.15.3.13 `std::string artdaq::CommanderInterface::send_status() [inline], [virtual]`

Using the transport mechanism, send a status command

The status command returns the current status of the artdaq process.

Returns

Command result: current status of the artdaq process

Definition at line 52 of file CommanderInterface.cc.

7.15.3.14 `std::string artdaq::CommanderInterface::send_stop(uint64_t, uint64_t) [inline], [virtual]`

Using the transport mechanism, send a stop command

The stop command stops the current Run. This command accepts a timeout parameter and a timestamp parameter.

Returns

Command result: "SUCCESS" if succeeded

Definition at line 40 of file CommanderInterface.cc.

7.15.3.15 `std::string artdaq::CommanderInterface::send_unregister_monitor(std::string) [inline], [virtual]`

Using the transport mechanism, send an unregister_monitor command

This will cause a Dispatcher to stop sending data to the monitor identified by the given label

Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc_commander](#).

Definition at line 76 of file CommanderInterface.cc.

7.15.4 Member Data Documentation

7.15.4.1 `artdaq::Commandable& artdaq::CommanderInterface::__commandable`

Reference to the [Commandable](#) that this Commander Commands.

Definition at line 193 of file CommanderInterface.hh.

The documentation for this class was generated from the following files:

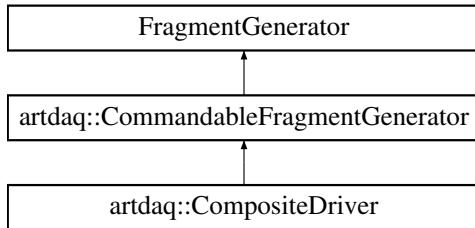
- artdaq/artdaq/ExternalComms/CommanderInterface.hh
- artdaq/artdaq/ExternalComms/CommanderInterface.cc

7.16 artdaq::CompositeDriver Class Reference

[CompositeDriver](#) handles a set of lower-level generators.

```
#include <artdaq/Application/CompositeDriver.hh>
```

Inheritance diagram for artdaq::CompositeDriver:



Public Member Functions

- [CompositeDriver \(fhicl::ParameterSet const &ps\)](#)
CompositeDriver Constructor.
- virtual ~[CompositeDriver \(\) noexcept](#)
Destructor. Calls the destructors for each configured CommandableFragmentGenerator.
- void [start \(\) override](#)
Start all configured CommandableFragmentGenerators.
- void [stopNoMutex \(\) override](#)
Call non-locked stop methods for all configured CommandableFragmentGenerators.
- void [stop \(\) override](#)
Call stop methods for all configured CommandableFragmentGenerators. Currently handled by stopNoMutex.
- void [pause \(\) override](#)
Pause all configured CommandableFragmentGenerators.
- void [resume \(\) override](#)
Resume all configured CommandableFragmentGenerators.

Additional Inherited Members

7.16.1 Detailed Description

[CompositeDriver](#) handles a set of lower-level generators.

When multiple CommandableFragmentGenerators are needed by a single BoardReader, [CompositeDriver](#) may be used to provide a single interface to all of them.

Definition at line 17 of file CompositeDriver.hh.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 artdaq::CompositeDriver::CompositeDriver (fhicl::ParameterSet const & ps) [explicit]

[CompositeDriver](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure CompositeDriver
-----------------	--

* CompositeDriver accepts the following Parameters:
 * "generator_config_list" (REQUIRED): A FHiCL sequence of FHiCL tables, each one configuring
 a CommandableFragmentGenerator instance.
 *

Definition at line 11 of file `CompositeDriver_generator.cc`.

The documentation for this class was generated from the following files:

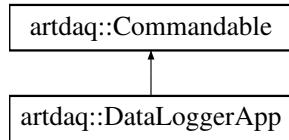
- `artdaq/artdaq/Application/CompositeDriver.hh`
- `artdaq/artdaq/Application/CompositeDriver_generator.cc`

7.17 `artdaq::DataLoggerApp` Class Reference

`DataLoggerApp` is an [artdaq::Commandable](#) derived class which controls the [DataLoggerCore](#).

```
#include <artdaq/Application/DataLoggerApp.hh>
```

Inheritance diagram for `artdaq::DataLoggerApp`:



Public Member Functions

- `DataLoggerApp` (int rank, std::string name)
DataLoggerApp Constructor.
- `DataLoggerApp` (`DataLoggerApp` const &)=delete
Copy Constructor is Deleted.
- virtual ~`DataLoggerApp` ()=default
Default virtual destructor.
- `DataLoggerApp` & `operator=` (`DataLoggerApp` const &)=delete
Copy Assignment operator is Deleted.
- bool `do_initialize` (fhicl::ParameterSet const &pset, uint64_t, uint64_t) override
Initialize the DataLoggerCore.
- bool `do_start` (art::RunID id, uint64_t, uint64_t) override
Start the DataLoggerCore.
- bool `do_stop` (uint64_t, uint64_t) override
Stop the DataLoggerCore.
- bool `do_pause` (uint64_t, uint64_t) override
Pause the DataLoggerCore.
- bool `do_resume` (uint64_t, uint64_t) override
Resume the DataLoggerCore.

- bool `do_shutdown` (uint64_t) override
Shutdown the DataLoggerCore.
- bool `do_soft_initialize` (fhicl::ParameterSet const &, uint64_t, uint64_t) override
Soft-initialize the DataLoggerCore. No-Op.
- bool `do_reinitialize` (fhicl::ParameterSet const &, uint64_t, uint64_t) override
Reinitialize the DataLoggerCore. No-Op.
- std::string `report` (std::string const &which) const override
If which is "transition_status", report the status of the last transition. Otherwise pass through to DataLoggerCore.

Additional Inherited Members

7.17.1 Detailed Description

`DataLoggerApp` is an `arddaq::Commandable` derived class which controls the `DataLoggerCore`.

Definition at line 17 of file `DataLoggerApp.hh`.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 artdaq::DataLoggerApp::DataLoggerApp (int rank, std::string name)

`DataLoggerApp` Constructor.

Parameters

<code>rank</code>	The rank of the DataLogger
<code>name</code>	The nickname of the DataLogger

Definition at line 7 of file `DataLoggerApp.cc`.

7.17.3 Member Function Documentation

7.17.3.1 bool artdaq::DataLoggerApp::do_initialize (fhicl::ParameterSet const & pset, uint64_t , uint64_t) [override], [virtual]

Initialize the `DataLoggerCore`.

Parameters

<code>pset</code>	ParameterSet used to initialize the <code>DataLoggerCore</code>
-------------------	---

Returns

Whether the initialize transition succeeded

Reimplemented from `arddaq::Commandable`.

Definition at line 16 of file `DataLoggerApp.cc`.

7.17.3.2 bool artdaq::DataLoggerApp::do_pause (uint64_t , uint64_t) [override], [virtual]

Pause the `DataLoggerCore`.

Returns

Whether the pause transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 65 of file DataLoggerApp.cc.

```
7.17.3.3 bool artdaq::DataLoggerApp::do_reinitialize( fhicl::ParameterSet const &, uint64_t , uint64_t ) [override],  
[virtual]
```

Reinitialize the [DataLoggerCore](#). No-Op.

Returns

This function always returns true

Reimplemented from [artdaq::Commandable](#).

Definition at line 109 of file DataLoggerApp.cc.

```
7.17.3.4 bool artdaq::DataLoggerApp::do_resume( uint64_t , uint64_t ) [override], [virtual]
```

Resume the [DataLoggerCore](#).

Returns

Whether the resume transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 78 of file DataLoggerApp.cc.

```
7.17.3.5 bool artdaq::DataLoggerApp::do_shutdown( uint64_t ) [override], [virtual]
```

Shutdown the [DataLoggerCore](#).

Returns

Whether the shutdown transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 91 of file DataLoggerApp.cc.

```
7.17.3.6 bool artdaq::DataLoggerApp::do_soft_initialize( fhicl::ParameterSet const &, uint64_t , uint64_t ) [override],  
[virtual]
```

Soft-initialize the [DataLoggerCore](#). No-Op.

Returns

This function always returns true

Reimplemented from [artdaq::Commandable](#).

Definition at line 104 of file DataLoggerApp.cc.

7.17.3.7 `bool artdaq::DataLoggerApp::do_start(art::RunID id, uint64_t, uint64_t) [override], [virtual]`

Start the [DataLoggerCore](#).

Parameters

<i>id</i>	Run number of the new run
-----------	---------------------------

Returns

Whether the start transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 36 of file DataLoggerApp.cc.

7.17.3.8 bool artdaq::DataLoggerApp::do_stop(uint64_t , uint64_t) [override], [virtual]

Stop the [DataLoggerCore](#).

Returns

Whether the stop transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 52 of file DataLoggerApp.cc.

7.17.3.9 DataLoggerApp& artdaq::DataLoggerApp::operator= (DataLoggerApp const &) [delete]

Copy Assignment operator is Deleted.

Returns

[DataLoggerApp](#) copy

7.17.3.10 std::string artdaq::DataLoggerApp::report(std::string const & which) const [override], [virtual]

If which is "transition_status", report the status of the last transition. Otherwise pass through to [DataLoggerCore](#).

Parameters

<i>which</i>	What to report on
--------------	-------------------

Returns

Report string. Empty for unknown "which" parameter

Reimplemented from [artdaq::Commandable](#).

Definition at line 114 of file DataLoggerApp.cc.

The documentation for this class was generated from the following files:

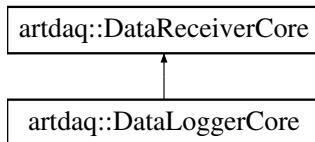
- artdaq/artdaq/Application/DataLoggerApp.hh
- artdaq/artdaq/Application/DataLoggerApp.cc

7.18 artdaq::DataLoggerCore Class Reference

[DataLoggerCore](#) implements the state machine for the DataLogger artdaq application. [DataLoggerCore](#) processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.

```
#include <artdaq/Application/DataLoggerCore.hh>
```

Inheritance diagram for artdaq::DataLoggerCore:



Public Member Functions

- [DataLoggerCore](#) (int rank, std::string name)
DataLoggerCore Constructor.
- [DataLoggerCore](#) ([DataLoggerCore](#) const &)=delete
Copy Constructor is deleted.
- [~DataLoggerCore](#) ()
- [DataLoggerCore](#) & [operator=](#) ([DataLoggerCore](#) const &)=delete
Copy Assignment operator is deleted.
- bool [initialize](#) (fhicl::ParameterSet const &pset) override
Processes the initialize request.

Additional Inherited Members

7.18.1 Detailed Description

[DataLoggerCore](#) implements the state machine for the DataLogger artdaq application. [DataLoggerCore](#) processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.

Definition at line 18 of file DataLoggerCore.hh.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 artdaq::DataLoggerCore::DataLoggerCore (int rank, std::string name)

[DataLoggerCore](#) Constructor.

Parameters

<i>rank</i>	Rank of the DataLogger
<i>name</i>	Friendly name for the DataLogger

Todo Make the global queue size configurable

Definition at line 21 of file DataLoggerCore.cc.

7.18.2.2 `artdaq::DataLoggerCore::~DataLoggerCore()`

Destructor.

Definition at line 26 of file DataLoggerCore.cc.

7.18.3 Member Function Documentation

7.18.3.1 `bool artdaq::DataLoggerCore::initialize(fhiCL::ParameterSet const & pset) [override], [virtual]`

Processes the initialize request.

Parameters

<code>pset</code>	ParameterSet used to configure the DataLoggerCore
-------------------	---

Returns

Whether the initialize attempt succeeded

```
* DataLoggerCore accepts the following Parameters:
* "daq" (REQUIRED): FHiCL table containing DAQ configuration
* "DataLogger" (REQUIRED): FHiCL table containing DataLogger paramters
*   "expected_events_per_bunch" (REQUIRED): Number of events to collect before sending them to art
*   "enq_timeout" (Default: 5.0): Maximum amount of time to wait while enqueueing events to the ConcurrentQueue
*   "is_data_logger": True if the DataLogger is a Data Logger
*   "is_online_monitor": True if the DataLogger is an Online Monitor. is_data_logger takes precedence
*   "is_dispatcher": True if the DataLogger is a Dispatcher. is_data_logger and is_online_monitor take precedence
*     NOTE: At least ONE of these three parameters must be specified.
*   "xmlrpc_client_list" (Default: ""): List of XMLRPC addresses of other applications in the artdaq system
*   "subrun_size_MB" (Default: 0): Maximum size for DataLogger-based subrun rollover
*   "subrun_duration" (Default: 0): Maximum time for DataLogger-based subrun rollover
*   "subrun_event_count" (Default: 0): Maximum event count for DataLogger-based subrun rollover
*   "inrun_recv_timeout_usec" (Default: 100000): Amount of time to wait for new events while running
*   "endrun_recv_timeout_usec" (Default: 20000000): Amount of time to wait for additional events at EndOfRun
*   "pause_recv_timeout_usec" (Default: 3000000): Amount of time to wait for additional events at PauseRun
*   "nonmon_event_prescale" (Default: 1): Only send 1/N events to art for online monitoring (requires is_data_logger)
*   "filesize_check_interval_seconds" (Default: 20): Interval to check the file size when using DataLogger-based
*   "filesize_check_interval_events" (Default: 20): Interval to check the file size when using DataLogger-based
*   "metrics": FHiCL table containing configuration for MetricManager
* "outputs" (REQUIRED): FHiCL table containing output parameters
*   "normalOutput" (REQUIRED): FHiCL table containing default output parameters
*     "fileName" (Default: ""): Name template of the output file. Used to determine output directory
*
```

Note that the "DataLogger" ParameterSet is also used to configure the EventStore. See that class' documentation for more information.

Implements [artdaq::DataReceiverCore](#).

Definition at line 31 of file DataLoggerCore.cc.

7.18.3.2 `DataLoggerCore& artdaq::DataLoggerCore::operator=(DataLoggerCore const &) [delete]`

Copy Assignment operator is deleted.

Returns

[DataLoggerCore](#) copy

The documentation for this class was generated from the following files:

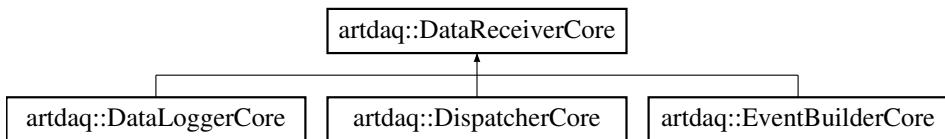
- artdaq/artdaq/Application/DataLoggerCore.hh
- artdaq/artdaq/Application/DataLoggerCore.cc

7.19 artdaq::DataReceiverCore Class Reference

[DataReceiverCore](#) implements the state machine for the DataReceiver artdaq application. [DataReceiverCore](#) receives Fragment objects from the [DataReceiverManager](#), and sends them to the EventStore.

```
#include <artdaq/Application/DataReceiverCore.hh>
```

Inheritance diagram for artdaq::DataReceiverCore:



Public Member Functions

- [DataReceiverCore](#) (int rank, std::string name)
DataReceiverCore Constructor.
- [DataReceiverCore](#) ([DataReceiverCore](#) const &)=delete
Copy Constructor is deleted.
- [~DataReceiverCore](#) ()
Copy Assignment operator is deleted.
- [DataReceiverCore](#) & [operator=](#) ([DataReceiverCore](#) const &)=delete
Copy Assignment operator is deleted.
- virtual bool [initialize](#) (fhicl::ParameterSet const &pset)=0
Processes the initialize request.
- bool [start](#) (art::RunID id)
Start the DataReceiverCore.
- bool [stop](#) ()
Stops the DataReceiverCore.
- bool [pause](#) ()
Pauses the DataReceiverCore.
- bool [resume](#) ()
Resumes the DataReceiverCore.
- bool [shutdown](#) ()
Shuts Down the DataReceiverCore.
- bool [soft_initialize](#) (fhicl::ParameterSet const &pset)
Soft-Initializes the DataReceiverCore. No-Op.
- bool [reinitialize](#) (fhicl::ParameterSet const &pset)
Reinitializes the DataReceiverCore.
- std::string [report](#) (std::string const &which) const
Send a report on a given run-time quantity.

Protected Member Functions

- bool `initializeDataReceiver` (fhicl::ParameterSet const &pset, fhicl::ParameterSet const &data_pset, fhicl::ParameterSet const &metric_pset)
Initialize the DataReceiverCore (should be called from initialize() overrides.
- void `logMessage_` (std::string const &text)
Log a message, setting severity based on verbosity flag.

Protected Attributes

- std::string `name_`
Name of this DataReceiverCore instance.
- std::unique_ptr<DataReceiverManager> `receiver_ptr_`
Pointer to the DataReceiverManager.
- std::shared_ptr<SharedMemoryEventManager> `event_store_ptr_`
Pointer to the SharedMemoryEventManager (art)
- std::atomic<bool> `stop_requested_`
Stop has been requested?
- std::atomic<bool> `pause_requested_`
Pause has been requested?
- std::atomic<bool> `run_is_paused_`
Pause has been successfully completed?
- bool `verbose_`
Whether to log verbosely.
- MetricManager `metricMan_`
MetricManager concrete instance (for Globals.hh::metricMan)

7.19.1 Detailed Description

`DataReceiverCore` implements the state machine for the DataReceiver artdaq application. `DataReceiverCore` receives Fragment objects from the `DataReceiverManager`, and sends them to the EventStore.

Definition at line 24 of file DataReceiverCore.hh.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 artdaq::DataReceiverCore::DataReceiverCore (int rank, std::string name)

`DataReceiverCore` Constructor.

Parameters

<code>rank</code>	Rank of the DataReceiver
-------------------	--------------------------

<i>name</i>	Friendly name for the DataReceiver
-------------	------------------------------------

Definition at line 14 of file DataReceiverCore.cc.

7.19.2.2 artdaq::DataReceiverCore::~DataReceiverCore ()

Destructor.

Definition at line 25 of file DataReceiverCore.cc.

7.19.3 Member Function Documentation

7.19.3.1 virtual bool artdaq::DataReceiverCore::initialize (fhiCL::ParameterSet const & pset) [pure virtual]

Processes the initialize request.

Parameters

<i>pset</i>	ParameterSet used to configure the DataReceiverCore
-------------	---

Returns

Whether the initialize attempt succeeded

- * DataReceiverCore accepts the following Parameters:
- * "daq" (REQUIRED): FHiCL table containing DAQ configuration
- * "event_builder" (REQUIRED): FHiCL table containing Aggregator paramters
- * "fragment_count" (REQUIRED): Number of Fragment objects to collect before sending them to art
- * "inrun_recv_timeout_usec" (Default: 100000): Amount of time to wait for new Fragment objects while running
- * "endrun_recv_timeout_usec" (Default: 20000000): Amount of time to wait for additional Fragment objects at EndRun
- * "pause_recv_timeout_usec" (Default: 3000000): Amount of time to wait for additional Fragment objects at Pause
- * "verbose" (Default: false): Whether to print more verbose status information
- * "metrics": FHiCL table containing configuration for MetricManager
- *

Note that the "event_builder" ParameterSet is also used to configure the [SharedMemoryEventManager](#). See that class' documentation for more information.

Implemented in [artdaq::DispatcherCore](#), [artdaq::DataLoggerCore](#), and [artdaq::EventBuilderCore](#).

7.19.3.2 bool artdaq::DataReceiverCore::initializeDataReceiver (fhiCL::ParameterSet const & pset, fhiCL::ParameterSet const & data_pset, fhiCL::ParameterSet const & metric_pset) [protected]

Initialize the [DataReceiverCore](#) (should be called from [initialize\(\)](#) overrides.

Parameters

<i>pset</i>	ParameterSet for art configuration
<i>data_pset</i>	ParameterSet for DataReceiverManager and SharedMemoryEventManager configuration
<i>metric_pset</i>	ParameterSet for MetricManager

Returns

Whether the initialize succeeded

Definition at line 30 of file DataReceiverCore.cc.

7.19.3.3 void artdaq::DataReceiverCore::logMessage_ (std::string const & *text*) [protected]

Log a message, setting severity based on verbosity flag.

Parameters

<i>text</i>	Message to log
-------------	----------------

Definition at line 242 of file DataReceiverCore.cc.

7.19.3.4 DataReceiverCore& artdaq::DataReceiverCore::operator= (DataReceiverCore const &) [delete]

Copy Assignment operator is deleted.

Returns

AggregatorCore copy

7.19.3.5 bool artdaq::DataReceiverCore::pause ()

Pauses the [DataReceiverCore](#).

Returns

True if no exception

Definition at line 128 of file DataReceiverCore.cc.

7.19.3.6 bool artdaq::DataReceiverCore::reinitialize (fhicl::ParameterSet const & pset)

Reinitializes the [DataReceiverCore](#).

Parameters

<i>pset</i>	ParameterSet for configuring DataReceiverCore
-------------	---

Returns

True if no exception

Definition at line 201 of file DataReceiverCore.cc.

7.19.3.7 std::string artdaq::DataReceiverCore::report (std::string const & which) const

Send a report on a given run-time quantity.

Parameters

<i>which</i>	Which quantity to report
--------------	--------------------------

Returns

A string containing the requested quantity.

report accepts the following values of "which": "event_count": The number of events received, or -1 if not initialized
"incomplete_event_count": The number of incomplete event bunches in the EventStore, or -1 if not initialized

Anything else will return the run number and an error message.

Definition at line 210 of file DataReceiverCore.cc.

7.19.3.8 bool artdaq::DataReceiverCore::resume ()

Resumes the [DataReceiverCore](#).

Returns

True if no exception

Definition at line 153 of file DataReceiverCore.cc.

7.19.3.9 bool artdaq::DataReceiverCore::shutdown ()

Shuts Down the [DataReceiverCore](#).

Returns

If the shutdown was successful

Definition at line 163 of file DataReceiverCore.cc.

7.19.3.10 bool artdaq::DataReceiverCore::soft_initialize (fhicl::ParameterSet const & pset)

Soft-Initializes the [DataReceiverCore](#). No-Op.

Parameters

<i>pset</i>	ParameterSet for configuring DataReceiverCore
-------------	---

Returns

Always returns true

Definition at line 193 of file DataReceiverCore.cc.

7.19.3.11 bool artdaq::DataReceiverCore::start (art::RunID id)

Start the [DataReceiverCore](#).

Parameters

<i>id</i>	Run ID of the current run
-----------	---------------------------

Returns

True if no exception

Definition at line 65 of file DataReceiverCore.cc.

7.19.3.12 bool artdaq::DataReceiverCore::stop ()

Stops the [DataReceiverCore](#).

Returns

True if no exception

Definition at line 78 of file DataReceiverCore.cc.

The documentation for this class was generated from the following files:

- artdaq/artdaq/Application/DataReceiverCore.hh
- artdaq/artdaq/Application/DataReceiverCore.cc

7.20 artdaq::DataReceiverManager Class Reference

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins Data-ReceiverMaanger runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

```
#include <artdaq/DAQrate/DataReceiverManager.hh>
```

Public Member Functions

- [DataReceiverManager](#) (const fhicl::ParameterSet &ps, std::shared_ptr< SharedMemoryEventManager > shm)
DataReceiverManager Constructor.
- virtual ~[DataReceiverManager](#) ()
DataReceiverManager Destructor.
- size_t [count](#) () const
Return the count of Fragment objects received by this DataReceiverManager.
- size_t [slotCount](#) (size_t rank) const
Get the count of Fragment objects received by this DataReceiverManager from a given source.
- size_t [byteCount](#) () const
Get the total size of all data received by this DataReceiverManager.
- void [start_threads](#) ()
Start receiver threads for all enabled sources.
- std::set< int > [enabled_sources](#) () const
Get the list of enabled sources.
- std::set< int > [running_sources](#) () const
Get the list of sources which are still receiving data.
- std::shared_ptr< SharedMemoryEventManager > [getSharedMemoryEventManager](#) () const
Get a handle to the SharedMemoryEventManager connected to this DataReceiverManager.
- std::shared_ptr< detail::FragCounter > [GetReceivedFragmentCount](#) ()
Get a pointer to the FragCounter instance tracking the number of received Fragments.

7.20.1 Detailed Description

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins Data-ReceiverMaanger runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

Definition at line 28 of file DataReceiverManager.hh.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 `artdaq::DataReceiverManager::DataReceiverManager (const fhicl::ParameterSet & ps, std::shared_ptr< SharedMemoryEventManager > shm) [explicit]`

[DataReceiverManager](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure the DataReceiverManager
<code>shm</code>	Pointer to SharedMemoryEventManager instance (destination for received data)

- * DataReceiverManager accepts the following Parameters:
- * "auto_suppression_enabled" (Default: true): Whether to suppress a source that gets too far ahead
- * "max_receive_difference" (Default: 50): Threshold (in sequence ID) for suppressing a source
- * "receive_timeout_usecs" (Default: 100000): The timeout for receive operations
- * "enabled_sources" (OPTIONAL): List of sources which are enabled. If not specified, all sources are assumed enabled
- * "sources" (Default: blank table): FHiCL table containing TransferInterface configurations for each source.
- * NOTE: "source_rank" MUST be specified (and unique) for each source!
- *

Definition at line 9 of file DataReceiverManager.cc.

7.20.3 Member Function Documentation

7.20.3.1 `size_t artdaq::DataReceiverManager::byteCount () const [inline]`

Get the total size of all data received by this [DataReceiverManager](#).

Returns

The total size of all data received by this [DataReceiverManager](#)

Definition at line 145 of file DataReceiverManager.hh.

7.20.3.2 `size_t artdaq::DataReceiverManager::count () const [inline]`

Return the count of Fragment objects received by this [DataReceiverManager](#).

Returns

The count of Fragment objects received by this [DataReceiverManager](#)

Definition at line 129 of file DataReceiverManager.hh.

7.20.3.3 `std::set<int> artdaq::DataReceiverManager::enabled_sources () const [inline]`

Get the list of enabled sources.

Returns

The list of enabled sources

Definition at line 82 of file DataReceiverManager.hh.

7.20.3.4 std::shared_ptr<detail::FragCounter> artdaq::DataReceiverManager::GetReceivedFragmentCount() [inline]

Get a pointer to the FragCounter instance tracking the number of received Fragments.

Returns

Pointer to the FragCounter instance tracking the number of received Fragments

Definition at line 101 of file DataReceiverManager.hh.

7.20.3.5 std::shared_ptr<SharedMemoryEventManager> artdaq::DataReceiverManager::getSharedMemoryEventManager() const [inline]

Get a handle to the [SharedMemoryEventManager](#) connected to this [DataReceiverManager](#).

Returns

shared_ptr to [SharedMemoryEventManager](#) instance

Definition at line 94 of file DataReceiverManager.hh.

7.20.3.6 std::set<int> artdaq::DataReceiverManager::running_sources() const [inline]

Get the list of sources which are still receiving data.

Returns

std::set containing ranks of sources which are still receiving data

Definition at line 88 of file DataReceiverManager.hh.

7.20.3.7 size_t artdaq::DataReceiverManager::slotCount(size_t rank) const [inline]

Get the count of Fragment objects received by this [DataReceiverManager](#) from a given source.

Parameters

<i>rank</i>	Source rank to get count for
-------------	------------------------------

Returns

The count of Fragment objects received by this [DataReceiverManager](#) from the source

Definition at line 137 of file DataReceiverManager.hh.

The documentation for this class was generated from the following files:

- artdaq/artdaq/DAQrate/DataReceiverManager.hh
- artdaq/artdaq/DAQrate/DataReceiverManager.cc

7.21 artdaq::DataSenderManager Class Reference

Sends Fragment objects using [TransferInterface](#) plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations.

```
#include <artdaq/DAQrate/DataSenderManager.hh>
```

Public Member Functions

- **DataSenderManager** (const fhiCL::ParameterSet &ps)

DataSenderManager Constructor.
- virtual ~**DataSenderManager** ()

DataSenderManager Destructor.
- int **sendFragment** (Fragment &&frag)

Send the given Fragment. Return the rank of the destination to which the Fragment was sent.
- size_t **count** () const

Return the count of Fragment objects sent by this DataSenderManager.
- size_t **slotCount** (size_t rank) const

Get the count of Fragment objects sent by this DataSenderManager to a given destination.
- size_t **destinationCount** () const

Get the number of configured destinations.
- std::set< int > **enabled_destinations** () const

Get the list of enabled destinations.
- size_t **GetRoutingTableEntryCount** () const

Gets the current size of the Routing Table, in case other parts of the system want to use this information.

7.21.1 Detailed Description

Sends Fragment objects using [TransferInterface](#) plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations.

Definition at line 26 of file DataSenderManager.hh.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 artdaq::DataSenderManager::DataSenderManager (const fhiCL::ParameterSet & ps) [explicit]

DataSenderManager Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure the DataSenderManager
-----------	--

```
* DataSenderManager accepts the following Parameters:
* "broadcast_sends" (Default: false): Send all Fragments to all destinations
* "nonblocking_sends" (Default: false): If true, will use non-reliable mode of TransferInterface plugins
* "routing_table_config" (Default: Empty ParameterSet): FHiCL table for RoutingMaster parameters
* "use_routing_master" (Default: false): True if using the Routing Master
* "table_update_port" (Default: 35556): Port that table updates should arrive on
* "table_update_address" (Default: "227.128.12.28"): Address that table updates should arrive on
* "table_acknowledge_port" (Default: 35557): Port that acknowledgements should be sent to
* "routing_master_hostname" (Default: "localhost"): Host that acknowledgements should be sent to
* "routing_timeout_ms" (Default: 1000): Time to wait for a routing table update if the table is exhausted
* "destinations" (Default: Empty ParameterSet): FHiCL table for TransferInterface configurations for each destination
* NOTE: "destination_rank" MUST be specified (and unique) for each destination!
* "enabled_destinations" (OPTIONAL): If specified, only the destination ranks listed will be enabled. If not specified, all destinations will be enabled.
```

Definition at line 14 of file DataSenderManager.cc.

7.21.3 Member Function Documentation

7.21.3.1 `size_t artdaq::DataSenderManager::count() const [inline]`

Return the count of Fragment objects sent by this DataSenderManager.

Returns

The count of Fragment objects sent by this [DataSenderManager](#)

Definition at line 137 of file DataSenderManager.hh.

7.21.3.2 `size_t artdaq::DataSenderManager::destinationCount() const [inline]`

Get the number of configured destinations.

Returns

The number of configured destinations

Definition at line 82 of file DataSenderManager.hh.

7.21.3.3 `std::set<int> artdaq::DataSenderManager::enabled_destinations() const [inline]`

Get the list of enabled destinations.

Returns

The list of enabled destination ranks

Definition at line 88 of file DataSenderManager.hh.

7.21.3.4 `size_t artdaq::DataSenderManager::GetRoutingTableEntryCount() const`

Gets the current size of the Routing Table, in case other parts of the system want to use this information.

Returns

The current size of the Routing Table. Note that the Routing Table is trimmed after each successful send.

Definition at line 258 of file DataSenderManager.cc.

7.21.3.5 `int artdaq::DataSenderManager::sendFragment(Fragment && frag)`

Send the given Fragment. Return the rank of the destination to which the Fragment was sent.

Parameters

<code>frag</code>	Fragment to sent
-------------------	------------------

Returns

Rank of destination for Fragment

Definition at line 312 of file DataSenderManager.cc.

7.21.3.6 `size_t artdaq::DataSenderManager::slotCount(size_t rank) const [inline]`

Get the count of Fragment objects sent by this [DataSenderManager](#) to a given destination.

Parameters

<code>rank</code>	Destination rank to get count for
-------------------	-----------------------------------

Returns

The count of Fragment objects sent by this [DataSenderManager](#) to the destination

Definition at line 145 of file DataSenderManager.hh.

The documentation for this class was generated from the following files:

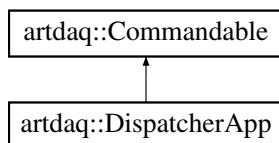
- artdaq/artdaq/DAQrate/DataSenderManager.hh
- artdaq/artdaq/DAQrate/DataSenderManager.cc

7.22 artdaq::DispatcherApp Class Reference

[DispatcherApp](#) is an [artdaq::Commandable](#) derived class which controls the [DispatcherCore](#).

```
#include <artdaq/Application/DispatcherApp.hh>
```

Inheritance diagram for artdaq::DispatcherApp:



Public Member Functions

- [DispatcherApp\(int rank, std::string name \)](#)
DispatcherApp Constructor.
- [DispatcherApp\(DispatcherApp const & \)=delete](#)
Copy Constructor is Deleted.
- [virtual ~DispatcherApp\(\)=default](#)
Default virtual destructor.
- [DispatcherApp & operator=\(DispatcherApp const & \)=delete](#)

- Copy Assignment operator is Deleted.*
- bool `do_initialize` (fhicl::ParameterSet const &pset, uint64_t, uint64_t) override
Initialize the `DispatcherCore`.
 - bool `do_start` (art::RunID id, uint64_t, uint64_t) override
Start the `DispatcherCore`.
 - bool `do_stop` (uint64_t, uint64_t) override
Stop the `DispatcherCore`.
 - bool `do_pause` (uint64_t, uint64_t) override
Pause the `DispatcherCore`.
 - bool `do_resume` (uint64_t, uint64_t) override
Resume the `DispatcherCore`.
 - bool `do_shutdown` (uint64_t) override
Shutdown the `DispatcherCore`.
 - bool `do_soft_initialize` (fhicl::ParameterSet const &, uint64_t, uint64_t) override
Soft-initialize the `DispatcherCore`. No-Op.
 - bool `do_reinitialize` (fhicl::ParameterSet const &, uint64_t, uint64_t) override
Reinitialize the `DispatcherCore`. No-Op.
 - std::string `report` (std::string const &which) const override
If which is "transition_status", report the status of the last transition. Otherwise pass through to `DispatcherCore`.
 - std::string `register_monitor` (fhicl::ParameterSet const &info) override
Register an art Online Monitor to the `DispatcherCore`.
 - std::string `unregister_monitor` (std::string const &label) override
Remove an art Online Monitor from the `DispatcherCore`.

Additional Inherited Members

7.22.1 Detailed Description

`DispatcherApp` is an `arddaq::Commandable` derived class which controls the `DispatcherCore`.

Definition at line 17 of file `DispatcherApp.hh`.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 artdaq::DispatcherApp::DispatcherApp (int rank, std::string name)

`DispatcherApp` Constructor.

Parameters

<code>rank</code>	The rank of the Dispatcher
<code>name</code>	The nickname of the Dispatcher

Definition at line 7 of file `DispatcherApp.cc`.

7.22.3 Member Function Documentation

7.22.3.1 bool artdaq::DispatcherApp::do_initialize (fhicl::ParameterSet const &pset, uint64_t , uint64_t) [override], [virtual]

Initialize the `DispatcherCore`.

Parameters

<i>pset</i>	ParameterSet used to initialize the DispatcherCore
-------------	--

Returns

Whether the initialize transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 16 of file DispatcherApp.cc.

7.22.3.2 bool artdaq::DispatcherApp::do_pause(uint64_t, uint64_t) [override], [virtual]

Pause the [DispatcherCore](#).

Returns

Whether the pause transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 65 of file DispatcherApp.cc.

7.22.3.3 bool artdaq::DispatcherApp::do_reinitialize(fhicl::ParameterSet const &, uint64_t, uint64_t) [override], [virtual]

Reinitialize the [DispatcherCore](#). No-Op.

Returns

This function always returns true

Reimplemented from [artdaq::Commandable](#).

Definition at line 108 of file DispatcherApp.cc.

7.22.3.4 bool artdaq::DispatcherApp::do_resume(uint64_t, uint64_t) [override], [virtual]

Resume the [DispatcherCore](#).

Returns

Whether the resume transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 77 of file DispatcherApp.cc.

7.22.3.5 bool artdaq::DispatcherApp::do_shutdown(uint64_t) [override], [virtual]

Shutdown the [DispatcherCore](#).

Returns

Whether the shutdown transition succeeded

Reimplemented from [arddaq::Commandable](#).

Definition at line 90 of file DispatcherApp.cc.

7.22.3.6 `bool artdaq::DispatcherApp::do_soft_initialize (fhicl::ParameterSet const &, uint64_t , uint64_t) [override], [virtual]`

Soft-initialize the [DispatcherCore](#). No-Op.

Returns

This function always returns true

Reimplemented from [arddaq::Commandable](#).

Definition at line 103 of file DispatcherApp.cc.

7.22.3.7 `bool artdaq::DispatcherApp::do_start (art::RunID id, uint64_t , uint64_t) [override], [virtual]`

Start the [DispatcherCore](#).

Parameters

<code>id</code>	Run number of the new run
-----------------	---------------------------

Returns

Whether the start transition succeeded

Reimplemented from [arddaq::Commandable](#).

Definition at line 36 of file DispatcherApp.cc.

7.22.3.8 `bool artdaq::DispatcherApp::do_stop (uint64_t , uint64_t) [override], [virtual]`

Stop the [DispatcherCore](#).

Returns

Whether the stop transition succeeded

Reimplemented from [arddaq::Commandable](#).

Definition at line 52 of file DispatcherApp.cc.

7.22.3.9 `DispatcherApp& artdaq::DispatcherApp::operator= (DispatcherApp const &) [delete]`

Copy Assignment operator is Deleted.

Returns

[DispatcherApp](#) copy

7.22.3.10 std::string artdaq::DispatcherApp::register_monitor (fhicl::ParameterSet const & *info*) [override],
[virtual]

Register an art Online Monitor to the [DispatcherCore](#).

Parameters

<i>info</i>	ParameterSet containing information about the monitor
-------------	---

Returns

String detailing result status

Reimplemented from [artdaq::Commandable](#).

Definition at line 146 of file DispatcherApp.cc.

7.22.3.11 std::string artdaq::DispatcherApp::report (std::string const & *which*) const [override], [virtual]

If *which* is "transition_status", report the status of the last transition. Otherwise pass through to [DispatcherCore](#).

Parameters

<i>which</i>	What to report on
--------------	-------------------

Returns

Report string. Empty for unknown "which" parameter

Reimplemented from [artdaq::Commandable](#).

Definition at line 113 of file DispatcherApp.cc.

7.22.3.12 std::string artdaq::DispatcherApp::unregister_monitor (std::string const & *label*) [override], [virtual]

Remove an art Online Monitor from the [DispatcherCore](#).

Parameters

<i>label</i>	Name of the monitor to remove
--------------	-------------------------------

Returns

String detailing result status

Reimplemented from [artdaq::Commandable](#).

Definition at line 171 of file DispatcherApp.cc.

The documentation for this class was generated from the following files:

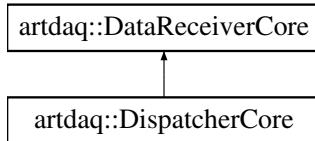
- artdaq/artdaq/Application/DispatcherApp.hh
- artdaq/artdaq/Application/DispatcherApp.cc

7.23 artdaq::DispatcherCore Class Reference

[DispatcherCore](#) implements the state machine for the Dispatcher artdaq application. [DispatcherCore](#) processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.

```
#include <artdaq/Application/DispatcherCore.hh>
```

Inheritance diagram for artdaq::DispatcherCore:



Public Member Functions

- [DispatcherCore](#) (int rank, std::string name)
DispatcherCore Constructor.
- [DispatcherCore](#) (DispatcherCore const &)=delete
Copy Constructor is deleted.
- [~DispatcherCore](#) ()
- [DispatcherCore & operator=](#) (DispatcherCore const &)=delete
Copy Assignment operator is deleted.
- bool [initialize](#) (fhicl::ParameterSet const &pset) override
Processes the initialize request.
- std::string [register_monitor](#) (fhicl::ParameterSet const &pset)
Create a new TransferInterface instance using the given configuration.
- std::string [unregister_monitor](#) (std::string const &label)
Delete the TransferInterface having the given unique label.

Additional Inherited Members

7.23.1 Detailed Description

[DispatcherCore](#) implements the state machine for the Dispatcher artdaq application. [DispatcherCore](#) processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.

Definition at line 21 of file DispatcherCore.hh.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 artdaq::DispatcherCore::DispatcherCore (int rank, std::string name)

[DispatcherCore](#) Constructor.

Parameters

<i>rank</i>	Rank of the Dispatcher
<i>name</i>	Friendly name for the Dispatcher

Todo Make the global queue size configurable

Definition at line 22 of file DispatcherCore.cc.

7.23.2.2 artdaq::DispatcherCore::~DispatcherCore()

Destructor.

Definition at line 26 of file DispatcherCore.cc.

7.23.3 Member Function Documentation

7.23.3.1 bool artdaq::DispatcherCore::initialize(fhiCL::ParameterSet const & pset) [override], [virtual]

Processes the initialize request.

Parameters

<i>pset</i>	ParameterSet used to configure the DispatcherCore
-------------	---

Returns

Whether the initialize attempt succeeded

```
* DispatcherCore accepts the following Parameters:
* "daq" (REQUIRED): FHiCL table containing DAQ configuration
* "Dispatcher" (REQUIRED): FHiCL table containing Dispatcher parameters
* "expected_events_per_bunch" (REQUIRED): Number of events to collect before sending them to art
* "enq_timeout" (Default: 5.0): Maximum amount of time to wait while enqueueing events to the ConcurrentQueue
* "is_data_logger": True if the Dispatcher is a Data Logger
* "is_online_monitor": True if the Dispatcher is an Online Monitor. is_data_logger takes precedence
* "is_dispatcher": True if the Dispatcher is a Dispatcher. is_data_logger and is_online_monitor take precedence
* NOTE: At least ONE of these three parameters must be specified.
* "xmlrpc_client_list" (Default: ""): List of XMLRPC addresses of other applications in the artdaq system
* "subrun_size_MB" (Default: 0): Maximum size for Dispatcher-based subrun rollover
* "subrun_duration" (Default: 0): Maximum time for Dispatcher-based subrun rollover
* "subrun_event_count" (Default: 0): Maximum event count for Dispatcher-based subrun rollover
* "inrun_recv_timeout_usec" (Default: 100000): Amount of time to wait for new events while running
* "endrun_recv_timeout_usec" (Default: 20000000): Amount of time to wait for additional events at EndOfRun
* "pause_recv_timeout_usec" (Default: 3000000): Amount of time to wait for additional events at PauseRun
* "onmon_event_prescale" (Default: 1): Only send 1/N events to art for online monitoring (requires is_data_logger)
* "filesize_check_interval_seconds" (Default: 20): Interval to check the file size when using Dispatcher-based
* "filesize_check_interval_events" (Default: 20): Interval to check the file size when using Dispatcher-based
* "metrics": FHiCL table containing configuration for MetricManager
* "outputs" (REQUIRED): FHiCL table containing output parameters
* "normalOutput" (REQUIRED): FHiCL table containing default output parameters
* "fileName" (Default: ""): Name template of the output file. Used to determine output directory
*
```

Note that the "Dispatcher" ParameterSet is also used to configure the EventStore. See that class' documentation for more information.

Implements [ardaq::DataReceiverCore](#).

Definition at line 31 of file DispatcherCore.cc.

7.23.3.2 DispatcherCore& artdaq::DispatcherCore::operator=(DispatcherCore const &) [delete]

Copy Assignment operator is deleted.

Returns

[DispatcherCore](#) copy

7.23.3.3 std::string artdaq::DispatcherCore::register_monitor (fhicl::ParameterSet const & pset)

Create a new [TransferInterface](#) instance using the given configuration.

Parameters

<i>pset</i>	ParameterSet used to configure the TransferInterface
-------------	--

Returns

String detailing any errors encountered or "Success"

See [TransferInterface](#) for details on the expected configuration

Definition at line 85 of file DispatcherCore.cc.

7.23.3.4 std::string artdaq::DispatcherCore::unregister_monitor (std::string const & *label*)

Delete the [TransferInterface](#) having the given unique label.

Parameters

<i>label</i>	Label of the TransferInterface to delete
--------------	--

Returns

String detailing any errors encountered or "Success"

Definition at line 154 of file DispatcherCore.cc.

The documentation for this class was generated from the following files:

- artdaq/artdaq/Application/DispatcherCore.hh
- artdaq/artdaq/Application/DispatcherCore.cc

7.24 art::detail::DummyProductCache Class Reference

A lightweight product cache for when the full thing is not appropriate (TBB and ROOT don't fully get along)

```
#include <artdaq/ArtModules/detail/DummyProductCache.h>
```

Public Member Functions

- EDProduct const * [product](#) (std::string const &wrappedName)
Retrieve a product handle from the cache.

7.24.1 Detailed Description

A lightweight product cache for when the full thing is not appropriate (TBB and ROOT don't fully get along)

Definition at line 16 of file DummyProductCache.h.

7.24.2 Member Function Documentation

7.24.2.1 EDProduct const * art::detail::DummyProductCache::product (std::string const & wrappedName)

Retrieve a product handle from the cache.

Parameters

<i>wrappedName</i>	Name of the product handle
--------------------	----------------------------

Returns

EDProduct instance corresponding to the given name

Definition at line 8 of file DummyProductCache.cc.

The documentation for this class was generated from the following files:

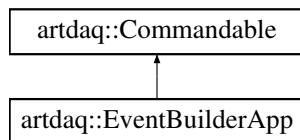
- artdaq/artdaq/ArtModules/detail/DummyProductCache.h
- artdaq/artdaq/ArtModules/detail/DummyProductCache.cc

7.25 artdaq::EventBuilderApp Class Reference

`EventBuilderApp` is an [artdaq::Commandable](#) derived class which controls the [EventBuilderCore](#).

```
#include <artdaq/Application/EventBuilderApp.hh>
```

Inheritance diagram for artdaq::EventBuilderApp:



Public Member Functions

- `EventBuilderApp (int rank, std::string name)`
EventBuilderApp Constructor.
- `EventBuilderApp (EventBuilderApp const &)=delete`
Copy Constructor is deleted.
- `virtual ~EventBuilderApp ()=default`
Default Destructor.
- `EventBuilderApp & operator= (EventBuilderApp const &)=delete`
Copy Assignment Operator is deleted.
- `bool do_initialize (fhicl::ParameterSet const &pset, uint64_t, uint64_t) override`
Initialize the EventBuilderCore.
- `bool do_start (art::RunID id, uint64_t, uint64_t) override`
Start the EventBuilderCore.
- `bool do_stop (uint64_t, uint64_t) override`
Stop the EventBuilderCore.
- `bool do_pause (uint64_t, uint64_t) override`
Pause the EventBuilderCore.
- `bool do_resume (uint64_t, uint64_t) override`
Resume the EventBuilderCore.
- `bool do_shutdown (uint64_t) override`

Shutdown the EventBuilderCore.

- bool `do_soft_initialize` (fhicl::ParameterSet const &pset, uint64_t, uint64_t) override

Soft-Initialize the EventBuilderCore.

- bool `do_reinitialize` (fhicl::ParameterSet const &pset, uint64_t, uint64_t) override

Reinitialize the EventBuilderCore.

- void `BootedEnter` () override

Action taken upon entering the "Booted" state.

- std::string `report` (std::string const &which) const override

If which is "transition_status", report the status of the last transition. Otherwise pass through to EventBuilderCore.

Additional Inherited Members

7.25.1 Detailed Description

`EventBuilderApp` is an `arddaq::Commandable` derived class which controls the `EventBuilderCore`.

Definition at line 17 of file `EventBuilderApp.hh`.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 artdaq::EventBuilderApp::EventBuilderApp (int rank, std::string name)

`EventBuilderApp` Constructor.

Parameters

<code>rank</code>	Rank of this EventBuilder
<code>name</code>	Friendly name of this application instance (MessageFacility Category)

Definition at line 3 of file `EventBuilderApp.cc`.

7.25.3 Member Function Documentation

7.25.3.1 void artdaq::EventBuilderApp::BootedEnter () [override], [virtual]

Action taken upon entering the "Booted" state.

This is a No-Op

Reimplemented from `arddaq::Commandable`.

Definition at line 128 of file `EventBuilderApp.cc`.

7.25.3.2 bool artdaq::EventBuilderApp::do_initialize (fhicl::ParameterSet const & pset, uint64_t , uint64_t) [override], [virtual]

Initialize the `EventBuilderCore`.

Parameters

<i>pset</i>	ParameterSet used to configure the EventBuilderCore
-------------	---

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 11 of file EventBuilderApp.cc.

7.25.3.3 bool artdaq::EventBuilderApp::do_pause (uint64_t , uint64_t) [override], [virtual]

Pause the [EventBuilderCore](#).

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 64 of file EventBuilderApp.cc.

7.25.3.4 bool artdaq::EventBuilderApp::do_reinitialize (fhicl::ParameterSet const & pset, uint64_t , uint64_t) [override], [virtual]

Reinitialize the [EventBuilderCore](#).

Parameters

<i>pset</i>	ParameterSet used to configure the EventBuilderCore
-------------	---

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 115 of file EventBuilderApp.cc.

7.25.3.5 bool artdaq::EventBuilderApp::do_resume (uint64_t , uint64_t) [override], [virtual]

Resume the [EventBuilderCore](#).

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 77 of file EventBuilderApp.cc.

7.25.3.6 `bool artdaq::EventBuilderApp::do_shutdown(uint64_t) [override], [virtual]`

Shutdown the [EventBuilderCore](#).

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 90 of file EventBuilderApp.cc.

7.25.3.7 `bool artdaq::EventBuilderApp::do_soft_initialize(fhicl::ParameterSet const & pset, uint64_t , uint64_t) [override], [virtual]`

Soft-Initialize the [EventBuilderCore](#).

Parameters

<code>pset</code>	ParameterSet used to configure the EventBuilderCore
-------------------	---

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 102 of file EventBuilderApp.cc.

7.25.3.8 `bool artdaq::EventBuilderApp::do_start(art::RunID id, uint64_t , uint64_t) [override], [virtual]`

Start the [EventBuilderCore](#).

Parameters

<code>id</code>	Run ID of new run
-----------------	-------------------

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 36 of file EventBuilderApp.cc.

7.25.3.9 `bool artdaq::EventBuilderApp::do_stop(uint64_t , uint64_t) [override], [virtual]`

Stop the [EventBuilderCore](#).

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 52 of file EventBuilderApp.cc.

7.25.3.10 EventBuilderApp& artdaq::EventBuilderApp::operator= (EventBuilderApp const &) [delete]

Copy Assignment Operator is deleted.

Returns

[EventBuilderApp](#) copy

7.25.3.11 std::string artdaq::EventBuilderApp::report (std::string const & which) const [override], [virtual]

If which is "transition_status", report the status of the last transition. Otherwise pass through to [EventBuilderCore](#).

Parameters

<i>which</i>	What to report on
--------------	-------------------

Returns

Report string. Empty for unknown "which" parameter

Reimplemented from [artdaq::Commandable](#).

Definition at line 139 of file EventBuilderApp.cc.

The documentation for this class was generated from the following files:

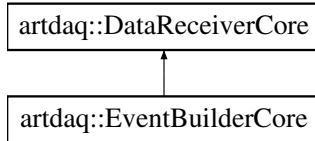
- artdaq/artdaq/Application/EventBuilderApp.hh
- artdaq/artdaq/Application/EventBuilderApp.cc

7.26 artdaq::EventBuilderCore Class Reference

[EventBuilderCore](#) implements the state machine for the EventBuilder artdaq application. [EventBuilderCore](#) receives Fragment objects from the [DataReceiverManager](#), and sends them to the EventStore.

```
#include <artdaq/Application/EventBuilderCore.hh>
```

Inheritance diagram for artdaq::EventBuilderCore:



Public Member Functions

- [EventBuilderCore](#) (int rank, std::string name)

EventBuilderCore Constructor.
- [EventBuilderCore](#) (EventBuilderCore const &)=delete

Copy Constructor is deleted.
- [~EventBuilderCore](#) ()

Destructor.
- [EventBuilderCore & operator=](#) (EventBuilderCore const &)=delete

- Copy Assignment operator is deleted.*
- bool `initialize` (fhiCL::ParameterSet const &pset) override
Processes the initialize request.

Additional Inherited Members

7.26.1 Detailed Description

`EventBuilderCore` implements the state machine for the EventBuilder artdaq application. `EventBuilderCore` receives Fragment objects from the `DataReceiverManager`, and sends them to the EventStore.

Definition at line 17 of file `EventBuilderCore.hh`.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 artdaq::EventBuilderCore::EventBuilderCore (int rank, std::string name)

`EventBuilderCore` Constructor.

Parameters

<code>rank</code>	Rank of the EventBuilder
<code>name</code>	Friendly name for the EventBuilder

Definition at line 14 of file `EventBuilderCore.cc`.

7.26.2.2 artdaq::EventBuilderCore::~EventBuilderCore ()

Destructor.

Definition at line 19 of file `EventBuilderCore.cc`.

7.26.3 Member Function Documentation

7.26.3.1 bool artdaq::EventBuilderCore::initialize (fhiCL::ParameterSet const & pset) [override], [virtual]

Processes the initialize request.

Parameters

<code>pset</code>	ParameterSet used to configure the <code>EventBuilderCore</code>
-------------------	--

Returns

Whether the initialize attempt succeeded

```
* EventBuilderCore accepts the following Parameters:
* "daq" (REQUIRED): FHiCL table containing DAQ configuration
* "event_builder" (REQUIRED): FHiCL table containing Aggregator parameters
* "fragment_count" (REQUIRED): Number of Fragment objects to collect before sending them to art
* "inrun_recv_timeout_usecs" (Default: 1000000): Amount of time to wait for new Fragment objects while running
* "endrun_recv_timeout_usecs" (Default: 20000000): Amount of time to wait for additional Fragment objects at EndRun
* "pause_recv_timeout_usecs" (Default: 3000000): Amount of time to wait for additional Fragment objects at Pause
* "verbose" (Default: false): Whether to print more verbose status information
* "metrics": FHiCL table containing configuration for MetricManager
```

Note that the "event_builder" ParameterSet is also used to configure the [SharedMemoryEventManager](#). See that class' documentation for more information.

Implements [artdaq::DataReceiverCore](#).

Definition at line 24 of file `EventBuilderCore.cc`.

7.26.3.2 `EventBuilderCore& artdaq::EventBuilderCore::operator= (EventBuilderCore const &) [delete]`

Copy Assignment operator is deleted.

Returns

`AggregatorCore copy`

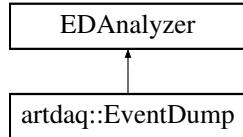
The documentation for this class was generated from the following files:

- `artdaq/artdaq/Application/EventBuilderCore.hh`
- `artdaq/artdaq/Application/EventBuilderCore.cc`

7.27 `artdaq::EventDump` Class Reference

Write Event information to the console.

Inheritance diagram for `artdaq::EventDump`:



Public Member Functions

- `EventDump (fhicl::ParameterSet const &pset)`

EventDump Constructor.

- virtual `~EventDump ()=default`

Default virtual Destructor.

- void `analyze (art::Event const &e) override`

This method is called for each art::Event in a file or run.

7.27.1 Detailed Description

Write Event information to the console.

Definition at line 32 of file `EventDump_module.cc`.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 `artdaq::EventDump::EventDump (fhicl::ParameterSet const & pset) [explicit]`

[EventDump](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure EventDump
-------------	--

* EventDump accepts the following Parameters:
 * "raw_data_label" (Default: "daq") : The label used to store artdaq data
 *

Definition at line 65 of file EventDump_module.cc.

7.27.3 Member Function Documentation

7.27.3.1 void artdaq::EventDump::analyze (art::Event const & e) [override]

This method is called for each art::Event in a file or run.

Parameters

<i>e</i>	The art::Event to analyze
----------	---------------------------

This module simply prints the event number, and art by default prints the products found in the event.

Definition at line 69 of file EventDump_module.cc.

The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/EventDump_module.cc

7.28 artdaq::detail::FragCounter Class Reference

Keep track of the count of Fragments received from a set of sources.

```
#include <artdaq/DAQrate/detail/FragCounter.hh>
```

Public Member Functions

- [FragCounter \(\)](#)
Default Constructor.
- [void incSlot \(size_t slot\)](#)
Increment the given slot by one.
- [void incSlot \(size_t slot, size_t inc\)](#)
Increment the given slot by the given amount.
- [void setSlot \(size_t slot, size_t val\)](#)
Set the given slot to the given value.
- [size_t nSlots \(\) const](#)
Get the number of slots this [FragCounter](#) instance is tracking.
- [size_t count \(\) const](#)
Get the total number of Fragments received.
- [size_t slotCount \(size_t slot\) const](#)
Get the current count for the requested slot.
- [size_t minCount \(\) const](#)
Get the minimum slot count.
- [size_t operator\[\] \(size_t slot\) const](#)
Get the current count for the requested slot.

7.28.1 Detailed Description

Keep track of the count of Fragments received from a set of sources.

Definition at line 20 of file FragCounter.hh.

7.28.2 Member Function Documentation

7.28.2.1 size_t artdaq::detail::FragCounter::count () const [inline]

Get the total number of Fragments received.

Returns

The total number of Fragments received

Definition at line 128 of file FragCounter.hh.

7.28.2.2 void artdaq::detail::FragCounter::incSlot (size_t slot) [inline]

Increment the given slot by one.

Parameters

<i>slot</i>	Slot to increment
-------------	-------------------

Definition at line 93 of file FragCounter.hh.

7.28.2.3 void artdaq::detail::FragCounter::incSlot (size_t slot, size_t inc) [inline]

Increment the given slot by the given amount.

Parameters

<i>slot</i>	Slot to increment
<i>inc</i>	Amount to increment

Definition at line 101 of file FragCounter.hh.

7.28.2.4 size_t artdaq::detail::FragCounter::minCount () const [inline]

Get the minimum slot count.

Returns

The minimum slot count

Definition at line 151 of file FragCounter.hh.

7.28.2.5 size_t artdaq::detail::FragCounter::nSlots () const [inline]

Get the number of slots this [FragCounter](#) instance is tracking.

Returns

The number of slots in this [FragCounter](#) instance

Definition at line 119 of file [FragCounter.hh](#).

7.28.2.6 size_t artdaq::detail::FragCounter::operator[](size_t slot) const [inline]

Get the current count for the requested slot.

Parameters

<i>slot</i>	Slot to get count for
-------------	-----------------------

Returns

The current count for the requested slot

Definition at line 78 of file [FragCounter.hh](#).

7.28.2.7 void artdaq::detail::FragCounter::setSlot(size_t slot, size_t val) [inline]

Set the given slot to the given value.

Parameters

<i>slot</i>	Slot to set
<i>val</i>	Value to set

Definition at line 110 of file [FragCounter.hh](#).

7.28.2.8 size_t artdaq::detail::FragCounter::slotCount(size_t slot) const [inline]

Get the current count for the requested slot.

Parameters

<i>slot</i>	Slot to get count for
-------------	-----------------------

Returns

The current count for the requested slot

Definition at line 142 of file [FragCounter.hh](#).

The documentation for this class was generated from the following file:

- artdaq/artdaq/DAQrate/detail/FragCounter.hh

7.29 artdaq::FragmentReceiverManager Class Reference

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins. Data-ReceiverManager runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

```
#include <proto/FragmentReceiverManager.hh>
```

Public Member Functions

- `FragmentReceiverManager (const fhicl::ParameterSet &ps)`
FragmentReceiverManager Constructor.
- `virtual ~FragmentReceiverManager ()`
FragmentReceiverManager Destructor.
- `FragmentPtr recvFragment (int &rank, size_t timeout_usec=0)`
Receive a Fragment.
- `size_t count () const`
Return the count of Fragment objects received by this FragmentReceiverManager.
- `size_t slotCount (size_t rank) const`
Get the count of Fragment objects received by this FragmentReceiverManager from a given source.
- `size_t byteCount () const`
Get the total size of all data received by this FragmentReceiverManager.
- `void start_threads ()`
Start receiver threads for all enabled sources.
- `std::set< int > enabled_sources () const`
Get the list of enabled sources.

7.29.1 Detailed Description

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins. Data-ReceiverMaanger runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

Definition at line 28 of file FragmentReceiverManager.hh.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 artdaq::FragmentReceiverManager::FragmentReceiverManager (`const fhicl::ParameterSet & ps`) [explicit]

[FragmentReceiverManager](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure the FragmentReceiverManager
-----------------	--

- * `FragmentReceiverManager` accepts the following Parameters:
- * `"auto_suppression_enabled"` (Default: true): Whether to suppress a source that gets too far ahead
- * `"max_receive_difference"` (Default: 50): Threshold (in sequence ID) for suppressing a source
- * `"receive_timeout_usec"` (Default: 100000): The timeout for receive operations
- * `"enabled_sources"` (OPTIONAL): List of sources which are enabled. If not specified, all sources are assumed enabled.
- * `"sources"` (Default: blank table): FHiCL table containing TransferInterface configurations for each source.
- * NOTE: `"source_rank"` MUST be specified (and unique) for each source!
- *

Definition at line 8 of file FragmentReceiverManager.cc.

7.29.3 Member Function Documentation

7.29.3.1 `size_t artdaq::FragmentReceiverManager::byteCount() const [inline]`

Get the total size of all data received by this [FragmentReceiverManager](#).

Returns

The total size of all data received by this [FragmentReceiverManager](#)

Definition at line 222 of file FragmentReceiverManager.hh.

7.29.3.2 `size_t artdaq::FragmentReceiverManager::count() const [inline]`

Return the count of Fragment objects received by this [FragmentReceiverManager](#).

Returns

The count of Fragment objects received by this [FragmentReceiverManager](#)

Definition at line 206 of file FragmentReceiverManager.hh.

7.29.3.3 `std::set<int> artdaq::FragmentReceiverManager::enabled_sources() const [inline]`

Get the list of enabled sources.

Returns

The list of enabled sources

Definition at line 89 of file FragmentReceiverManager.hh.

7.29.3.4 `artdaq::FragmentPtr artdaq::FragmentReceiverManager::recvFragment(int & rank, size_t timeout_usec = 0)`

Receive a Fragment.

Parameters

<code>out</code>	<code>rank</code>	Rank of sender that sent the Fragment, or RECV_TIMEOUT
	<code>timeout_usec</code>	Timeout to wait for a Fragment to become ready

Returns

Pointer to received Fragment. May be nullptr if no Fragments are ready

Definition at line 113 of file FragmentReceiverManager.cc.

7.29.3.5 `size_t artdaq::FragmentReceiverManager::slotCount(size_t rank) const [inline]`

Get the count of Fragment objects received by this [FragmentReceiverManager](#) from a given source.

Parameters

<code>rank</code>	Source rank to get count for
-------------------	------------------------------

Returns

The count of Fragment objects received by this [FragmentReceiverManager](#) from the source

Definition at line 214 of file FragmentReceiverManager.hh.

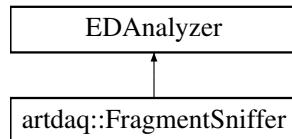
The documentation for this class was generated from the following files:

- artdaq/proto/FragmentReceiverManager.hh
- artdaq/proto/FragmentReceiverManager.cc

7.30 artdaq::FragmentSniffer Class Reference

This art::EDAnalyzer plugin tries to get Fragments from each event, asserting that the correct number of Fragments were present.

Inheritance diagram for artdaq::FragmentSniffer:



Public Member Functions

- [FragmentSniffer \(fhicl::ParameterSet const &p\)](#)

FragmentSniffer Constructor.

- virtual [~FragmentSniffer \(\)=default](#)

Default destructor.

- void [analyze \(art::Event const &e\) override](#)

Called for each event. Asserts that Fragment objects are present in the event and that the correct number of Fragments were found.

- void [endJob \(\) override](#)

Called at the end of the job. Asserts that the number of events processed was equal to the expected number.

7.30.1 Detailed Description

This art::EDAnalyzer plugin tries to get Fragments from each event, asserting that the correct number of Fragments were present.

Definition at line 21 of file FragmentSniffer_module.cc.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 `artdaq::FragmentSniffer::FragmentSniffer (fhicl::ParameterSet const & p) [explicit]`

`FragmentSniffer` Constructor.

Parameters

<i>p</i>	ParameterSet used to configure FragmentSniffer
----------	--

* FragmentSniffer accepts the following Parameters:
 * "raw_label" (Default: "daq"): Label under which Fragments are stored
 * "product_instance_name" (REQUIRED): Instance name under which Fragments are stored (Should be Fragment type name)
 * "num frags_per_event" (REQUIRED): Expected number of Fragments in each event
 * "num_events_expected" (Default: 0): Expected number of events in the job. If 0, will not perform end-of-job test
 *

Definition at line 62 of file FragmentSniffer_module.cc.

7.30.3 Member Function Documentation

7.30.3.1 void artdaq::FragmentSniffer::analyze (art::Event const & e) [override]

Called for each event. Asserts that Fragment objects are present in the event and that the correct number of Fragments were found.

Parameters

<i>e</i>	Event to analyze
----------	------------------

Definition at line 70 of file FragmentSniffer_module.cc.

The documentation for this class was generated from the following file:

- artdaq/test/ArtModules/FragmentSniffer_module.cc

7.31 artdaq::FragmentStoreElement Class Reference

This class contains tracking information for all Fragment objects which have been received from a specific source.

```
#include <proto/FragmentReceiverManager.hh>
```

Public Member Functions

- [FragmentStoreElement \(\)](#)
FragmentStoreElement Constructor.
- bool [empty \(\) const](#)
Are any Fragment objects contained in this FragmentStoreElement?
- void [emplace_front \(FragmentPtr &&frag\)](#)
Add a Fragment to the front of the FragmentStoreElement.
- void [emplace_back \(FragmentPtr &&frag\)](#)
Add a Fragment to the end of the FragmentStoreElement.
- FragmentPtr [front \(\)](#)
Remove the first Fragment from the FragmentStoreElement and return it.
- void [SetEndOfData \(size_t eod\)](#)
Set the End-Of-Data marker value for this Receiver.
- size_t [GetEndOfData \(\) const](#)
Get the value of the End-Of-Data marker for this Receiver.

7.31.1 Detailed Description

This class contains tracking information for all Fragment objects which have been received from a specific source.

This class was designed so that there could be a mutex for each source, instead of locking all sources whenever a Fragment had to be retrieved. [FragmentStoreElement](#) is itself a container type, sorted by Fragment arrival time. It is a modified queue, with only the first element accessible, but it allows elements to be added to either end (for rejected Fragments).

Definition at line 127 of file FragmentReceiverManager.hh.

7.31.2 Member Function Documentation

7.31.2.1 void artdaq::FragmentStoreElement::emplace_back (FragmentPtr && *frag*) [inline]

Add a Fragment to the end of the [FragmentStoreElement](#).

Parameters

<i>frag</i>	Fragment to add
-------------	-----------------

Definition at line 165 of file FragmentReceiverManager.hh.

7.31.2.2 void artdaq::FragmentStoreElement::emplace_front (FragmentPtr && *frag*) [inline]

Add a Fragment to the front of the [FragmentStoreElement](#).

Parameters

<i>frag</i>	Fragment to add
-------------	-----------------

Definition at line 154 of file FragmentReceiverManager.hh.

7.31.2.3 bool artdaq::FragmentStoreElement::empty () const [inline]

Are any Fragment objects contained in this [FragmentStoreElement](#)?

Returns

Whether any Fragment objects are contained in this [FragmentStoreElement](#)

Definition at line 145 of file FragmentReceiverManager.hh.

7.31.2.4 FragmentPtr artdaq::FragmentStoreElement::front () [inline]

Remove the first Fragment from the [FragmentStoreElement](#) and return it.

Returns

The first Fragment in the [FragmentStoreElement](#)

Definition at line 176 of file FragmentReceiverManager.hh.

7.31.2.5 `size_t artdaq::FragmentStoreElement::GetEndOfData() const [inline]`

Get the value of the End-Of-Data marker for this Receiver.

Returns

The value of the End-Of-Data marker. Returns -1 (0xFFFFFFFFFFFFFF) if no EndOfData Fragments received

Definition at line 194 of file FragmentReceiverManager.hh.

7.31.2.6 `void artdaq::FragmentStoreElement::SetEndOfData(size_t eod) [inline]`

Set the End-Of-Data marker value for this Receiver.

Parameters

<code>eod</code>	Number of Receives expected for this receiver
------------------	---

Definition at line 189 of file FragmentReceiverManager.hh.

The documentation for this class was generated from the following file:

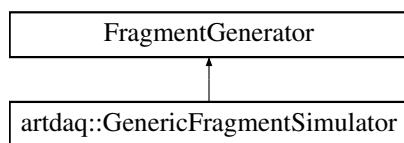
- artdaq/proto/FragmentReceiverManager.hh

7.32 artdaq::GenericFragmentSimulator Class Reference

[GenericFragmentSimulator](#) creates simulated Generic events, with data distributed according to a "histogram" provided in the configuration data.

```
#include <artdaq/DAQdata/GenericFragmentSimulator.hh>
```

Inheritance diagram for artdaq::GenericFragmentSimulator:



Public Types

- enum `content_selector_t` : `uint8_t` { `content_selector_t::EMPTY` = 0, `content_selector_t::FRAG_ID` = 1, `content_selector_t::RANDOM` = 2, `content_selector_t::DEAD_BEEF` }

What type of content should the [GenericFragmentSimulator](#) put in Fragment objects?

Public Member Functions

- [`GenericFragmentSimulator`](#) (`fhicl::ParameterSet const &ps`)
GenericFragmentSimulator Constructor.
- `bool getNext(Fragment::sequence_id_t sequence_id, Fragment::fragment_id_t fragment_id, FragmentPtr &frag_ptr)`
Generate a Fragment according to the value of the `content_selector_t` enum.

- bool `getNext` (FragmentPtrs &output) override
Get the next Fragment from the generator.
- std::vector
`< Fragment::fragment_id_t > fragmentIDs ()` override
Get the Fragment IDs generated by this instance.

7.32.1 Detailed Description

[GenericFragmentSimulator](#) creates simulated Generic events, with data distributed according to a "histogram" provided in the configuration data.

With this implementation, a single call to `getNext(frags)` will return a complete event (event ids are incremented automatically); fragment ids are sequential. Event size and content are both configurable; see the implementation for details.

Definition at line 26 of file `GenericFragmentSimulator.hh`.

7.32.2 Member Enumeration Documentation

7.32.2.1 enum artdaq::GenericFragmentSimulator::content_selector_t : uint8_t [strong]

What type of content should the [GenericFragmentSimulator](#) put in Fragment objects?

Enumerator

- EMPTY** Nothing (Default-initialized Fragment)
- FRAG_ID** Fill the payload with the Fragment ID.
- RANDOM** Use a random distribution to fill the payload.
- DEAD_BEEF** Fill the payload with 0xDEADBEEFDEADBEEF.

Definition at line 53 of file `GenericFragmentSimulator.hh`.

7.32.3 Constructor & Destructor Documentation

7.32.3.1 artdaq::GenericFragmentSimulator::GenericFragmentSimulator (fhicl::ParameterSet const & ps) [explicit]

[GenericFragmentSimulator](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure the GenericFragmentSimulator
-----------------	---

- * GenericFragmentSimulator accepts the following Parameters:
- * "content_selection" (Default: 0): What type of data to fill in generated Fragment payloads
 - * 0: Use uninitialized memory
 - * 1: Use the Fragment ID
 - * 2: Use random data
 - * 3: Use the word 0xDEADBEEFDEADBEEF
- * "payload_size" (Default: 10240): The size (in words) of the Fragment payload
- * "want_random_payload_size" (Default: false): Whether payload size should be sampled from a random distribution
- * "random_seed" (Default: 314159): Random seed for random number distributions
- * "fragments_per_event" (Default: 5): The number of Fragment objects to generate for each sequence ID
- * "starting_fragment_id" (Default: 0): The first Fragment ID handled by this GenericFragmentSimulator.
 - * Fragment IDs will be starting_fragment_id to starting_fragment_id + fragments_per_event.
- *

Definition at line 10 of file GenericFragmentSimulator_generator.cc.

7.32.4 Member Function Documentation

7.32.4.1 std::vector<Fragment::fragment_id_t> artdaq::GenericFragmentSimulator::fragmentIDs() [inline], [override]

Get the Fragment IDs generated by this instance.

Returns

The Fragment IDs generated by this instance

Definition at line 89 of file GenericFragmentSimulator.hh.

7.32.4.2 bool artdaq::GenericFragmentSimulator::getNext(Fragment::sequence_id_t sequence_id, Fragment::fragment_id_t fragment_id, FragmentPtr & frag_ptr)

Generate a Fragment according to the value of the content_selector_t enum.

Parameters

	<i>sequence_id</i>	Sequence ID of generated Fragment
	<i>fragment_id</i>	Fragment ID of generated Fragment
out	<i>frag_ptr</i>	Generated Fragment

Returns

True if no exception or assertion failure

Definition at line 45 of file GenericFragmentSimulator_generator.cc.

7.32.4.3 bool artdaq::GenericFragmentSimulator::getNext(FragmentPtrs & output) [inline], [override]

Get the next Fragment from the generator.

Parameters

out	<i>output</i>	List of FragmentPtr objects to add the new Fragment to
-----	---------------	--

Returns

Whether data taking should continue

Definition at line 80 of file GenericFragmentSimulator.hh.

The documentation for this class was generated from the following files:

- artdaq/artdaq/DAQdata/GenericFragmentSimulator.hh
- artdaq/artdaq/DAQdata/GenericFragmentSimulator_generator.cc

7.33 artdaq::GetPackageBuildInfo Struct Reference

Wrapper around the `artdaq::GetPackageBuildInfo::getPackageBuildInfo` function.

```
#include <artdaq/BuildInfo/GetPackageBuildInfo.hh>
```

Static Public Member Functions

- static `artdaq::PackageBuildInfo getPackageBuildInfo ()`

Gets the version number and build timestamp for artdaq.

7.33.1 Detailed Description

Wrapper around the `artdaq::GetPackageBuildInfo::getPackageBuildInfo` function.

Definition at line 17 of file `GetPackageBuildInfo.hh`.

7.33.2 Member Function Documentation

7.33.2.1 static `artdaq::PackageBuildInfo artdaq::GetPackageBuildInfo::getPackageBuildInfo () [static]`

Gets the version number and build timestamp for artdaq.

Returns

An `artdaq::PackageBuildInfo` object containing the version number and build timestamp for artdaq

The documentation for this struct was generated from the following file:

- `artdaq/artdaq/BuildInfo/GetPackageBuildInfo.hh`

7.34 artdaq::Globals Class Reference

The `artdaq::Globals` class contains several variables which are useful across the entire artdaq system.

```
#include <artdaq/DAQdata/Globals.hh>
```

Static Public Member Functions

- static double `timevalAsDouble (struct timeval tv)`

Convert a timeval value to a double.

- static uint32_t `seedAndRandom_ ()`

Seed the C random number generator with the current time (if that has not been done already) and generate a random value.

Static Public Attributes

- static int `my_rank_` = -1
The rank of the current application.
- static MetricManager * `metricMan_` = nullptr
A handle to MetricManager.

7.34.1 Detailed Description

The `artdaq::Globals` class contains several variables which are useful across the entire artdaq system.

Definition at line 28 of file `Globals.hh`.

7.34.2 Member Function Documentation

7.34.2.1 static uint32_t artdaq::Globals::seedAndRandom_() [inline], [static]

Seed the C random number generator with the current time (if that has not been done already) and generate a random value.

Returns

A random number.

Definition at line 45 of file `Globals.hh`.

7.34.2.2 double artdaq::Globals::timevalAsDouble (struct timeval tv) [static]

Convert a timeval value to a double.

Parameters

<code>tv</code>	Timeval to convert
-----------------	--------------------

Returns

timeval represented as a double

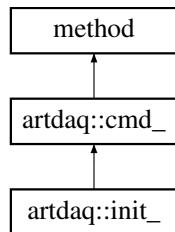
Definition at line 6 of file `Globals.cc`.

The documentation for this class was generated from the following files:

- `artdaq/artdaq/DAQdata/Globals.hh`
- `artdaq/artdaq/DAQdata/Globals.cc`

7.35 artdaq::init_ Class Reference

Inheritance diagram for `artdaq::init_`:



Public Member Functions

- [init_\(xmlrpc_commander &c\)](#)

Static Public Attributes

- static const uint64_t [defaultTimeout](#) = 45
- static const uint64_t [defaultTimestamp](#) = std::numeric_limits<const uint64_t>::max()

Additional Inherited Members

7.35.1 Detailed Description

Command class representing an init transition

Definition at line 389 of file `xmlrpc_commander.cc`.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 `artdaq::init_::init_(xmlrpc_commander &c)` [inline], [explicit]

Command class Constructor *

Parameters

<code>c</code>	xmlrpc_commander to send parsed command to \
----------------	--

Definition at line 389 of file `xmlrpc_commander.cc`.

7.35.3 Member Data Documentation

7.35.3.1 `const uint64_t artdaq::init_::defaultTimeout = 45` [static]

Default timeout for command

Definition at line 389 of file `xmlrpc_commander.cc`.

7.35.3.2 `const uint64_t artdaq::init_::defaultTimestamp = std::numeric_limits<const uint64_t>::max()` [static]

Default timestamp for Command

Definition at line 389 of file `xmlrpc_commander.cc`.

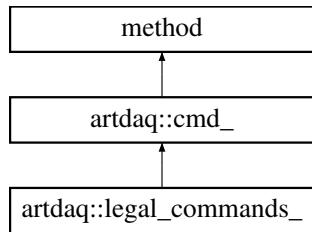
The documentation for this class was generated from the following file:

- artdaq/artdaq/ExternalComms/xmlrpc_commander.cc

7.36 artdaq::legal_commands_ Class Reference

[legal_commands_ Command class](#)

Inheritance diagram for artdaq::legal_commands_:



Public Member Functions

- [legal_commands_\(xmlrpc_commander &c\)](#)
legal_commands_ Constructor

Additional Inherited Members

7.36.1 Detailed Description

[legal_commands_ Command class](#)

Definition at line 567 of file `xmlrpc_commander.cc`.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 artdaq::legal_commands_::legal_commands_(xmlrpc_commander & c) [inline]

[legal_commands_ Constructor](#)

Parameters

c	<code>xmlrpc_commander</code> to send transition commands to
---	--

Definition at line 574 of file `xmlrpc_commander.cc`.

The documentation for this class was generated from the following file:

- artdaq/artdaq/ExternalComms/xmlrpc_commander.cc

7.37 LockFile Class Reference

Create a "lock file", removing it upon class destruction.

Public Member Functions

- [LockFile](#) (std::string const &path)
Create a lock file with the given path.
- [~LockFile](#) ()
LockFile Destructor, removes the lock file.

Static Public Member Functions

- static bool [IsLocked](#) (std::string const &path)
Check if the given lock file exists.

7.37.1 Detailed Description

Create a "lock file", removing it upon class destruction.

Definition at line 42 of file routing_master.cc.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 LockFile::LockFile(std::string const & path) [inline], [explicit]

Create a lock file with the given path.

Parameters

<i>path</i>	Path to lock file
-------------	-------------------

Definition at line 49 of file routing_master.cc.

7.37.3 Member Function Documentation

7.37.3.1 static bool LockFile::IsLocked(std::string const & path) [inline], [static]

Check if the given lock file exists.

Parameters

<i>path</i>	Path to lock file
-------------	-------------------

Returns

Whether the lock file exists, as determined by boost::filesystem

Definition at line 67 of file routing_master.cc.

The documentation for this class was generated from the following file:

- artdaq/proto/routing_master.cc

7.38 MessHead Struct Reference

This header is sent by the TCPSocket_transfer to allow for more efficient writev calls.

```
#include <artdaq/TransferPlugins/detail/SRSocket.h>
```

Public Types

- enum **MessType** {
 connect_v0 = 0, **data_v0**, **data_more_v0**, **stop_v0**,
routing_v0 }

The Message Type.

Public Attributes

- **uint8_t endian**
0=little(intel), 1=big
- **MessType message_type**
Message Type.
- **int64_t source_id**
Rank of the source.
- union {
 uint32_t conn_magic
*unsigned first is better for **MessHead** initializer: {0,0,my_node_idx_,CONN_MAGIC}*
int32_t byte_count
use CONN_MAGIC for connect_v0, data that follow for data_v0 (and 0 lenght data)
 };

7.38.1 Detailed Description

This header is sent by the TCPSocket_transfer to allow for more efficient writev calls.

Definition at line 15 of file SRSocket.h.

7.38.2 Member Enumeration Documentation

7.38.2.1 enum **MessHead::MessType**

The Message Type.

Only add to the end!

Definition at line 24 of file SRSocket.h.

The documentation for this struct was generated from the following file:

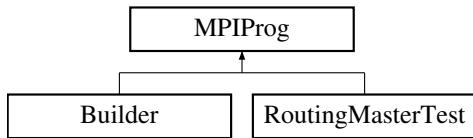
- artdaq/artdaq/TransferPlugins/detail/SRSocket.h

7.39 MPIProg Struct Reference

A wrapper for a MPI program. Similar to MPISentry.

```
#include <proto/MPIProg.hh>
```

Inheritance diagram for MPIProg:



Public Member Functions

- [MPIProg](#) (int argc, char **argv)

MPIProg Constructor.

- [~MPIProg](#) ()

MPIProg destructor.

Public Attributes

- int [procs_](#)

Number of processes in MPI cluster.

7.39.1 Detailed Description

A wrapper for a MPI program. Similar to MPISentry.

Definition at line 10 of file MPIProg.hh.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 [MPIProg::MPIProg](#) (int *argc*, char ** *argv*) [inline]

[MPIProg](#) Constructor.

Parameters

<i>argc</i>	Argument count
<i>argv</i>	Array of arguments as strings

Definition at line 17 of file MPIProg.hh.

The documentation for this struct was generated from the following file:

- artdaq/proto/MPIProg.hh

7.40 artdaq::MPISentry Class Reference

The [MPISentry](#) class initializes and finalizes the MPI context that the artdaq applications run in.

```
#include <artdaq/Application/MPI2/MPISentry.hh>
```

Public Member Functions

- [MPISentry](#) (int *argc_ptr, char ***argv_ptr)
MPISentry Constructor.
- [MPISentry](#) (int *argc_ptr, char ***argv_ptr, int [threading_level](#))
MPISentry Constructor.
- [MPISentry](#) (int *argc_ptr, char ***argv_ptr, int [threading_level](#), artdaq::TaskType type, MPI_Comm &local_group_comm)
MPISentry Constructor.
- [~MPISentry](#) ()
MPISentry Destructor. Calls MPI_Finalize.
- int [threading_level](#) () const
Get the actual threading level.
- int [rank](#) () const
Get the MPI rank of the application.
- int [procs](#) () const
The number of processes in the MPI context.

7.40.1 Detailed Description

The [MPISentry](#) class initializes and finalizes the MPI context that the artdaq applications run in.

Definition at line 15 of file MPISentry.hh.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 artdaq::MPISentry::MPISentry (int * argc_ptr, char *** argv_ptr)

[MPISentry](#) Constructor.

Parameters

<i>argc_ptr</i>	Pointer to the main argc
<i>argv_ptr</i>	Pointer to the main argv

Definition at line 9 of file MPISentry.cc.

7.40.2.2 artdaq::MPISentry::MPISentry (int * argc_ptr, char *** argv_ptr, int [threading_level](#))

[MPISentry](#) Constructor.

Parameters

<i>argc_ptr</i>	Pointer to the main argc
<i>argv_ptr</i>	Pointer to the main argv
<i>threading_level</i>	Requested MPI threading level

Definition at line 20 of file MPISentry.cc.

7.40.2.3 `artdaq::MPISentry::MPISentry (int * argc_ptr, char *** argv_ptr, int threading_level, artdaq::TaskType type, MPI_Comm & local_group_comm)`

[MPISentry](#) Constructor.

Parameters

<i>argc_ptr</i>	Pointer to the main argc
<i>argv_ptr</i>	Pointer to the main argv
<i>threading_level</i>	Requested MPI threading level
<i>type</i>	The application type of this application.
<i>local_group_comm</i>	The local communication for this application

Definition at line 48 of file MPISentry.cc.

7.40.3 Member Function Documentation

7.40.3.1 `int artdaq::MPISentry::procs () const`

The number of processes in the MPI context.

Returns

The number of processes in the MPI context

Definition at line 120 of file MPISentry.cc.

7.40.3.2 `int artdaq::MPISentry::rank () const`

Get the MPI rank of the application.

Returns

The MPI rank of the application

Definition at line 113 of file MPISentry.cc.

7.40.3.3 `int artdaq::MPISentry::threading_level () const`

Get the actual threading level.

Returns

The actual threading level provided by MPI

Definition at line 106 of file MPISentry.cc.

The documentation for this class was generated from the following files:

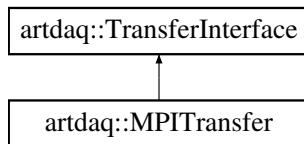
- artdaq/artdaq/Application/MPI2/MPISentry.hh
- artdaq/artdaq/Application/MPI2/MPISentry.cc

7.41 artdaq::MPITransfer Class Reference

[MPITransfer](#) is a [TransferInterface](#) implementation plugin that transfers data using MPI.

```
#include <artdaq/TransferPlugins/MPITransfer.hh>
```

Inheritance diagram for artdaq::MPITransfer:



Public Member Functions

- [**MPITransfer**](#) (fhicl::ParameterSet pset, [Role role](#))
MPITransfer Constructor.
- [**MPITransfer**](#) (const [MPITransfer](#) &)=delete
Copy Constructor is deleted.
- [**MPITransfer** & **operator=**](#) (const [MPITransfer](#) &)=delete
Copy Assignment operator is deleted.
- virtual [**~MPITransfer**](#) ()
MPITransfer Destructor.
- [**CopyStatus copyFragment**](#) (Fragment &frag, size_t timeout_usec=std::numeric_limits< size_t >::max()) override
Copy a Fragment to the destination. Forces asynchronous send.
- [**CopyStatus moveFragment**](#) (Fragment &&frag, size_t timeout_usec=std::numeric_limits< size_t >::max()) override
Move a Fragment to the destination.
- int [**receiveFragmentHeader**](#) (detail::RawFragmentHeader &header, size_t receiveTimeout) override
Receive a Fragment Header from the transport mechanism.
- int [**receiveFragmentData**](#) (RawDataType *destination, size_t wordCount) override
Receive the body of a Fragment to the given destination pointer.

Additional Inherited Members

7.41.1 Detailed Description

[MPITransfer](#) is a [TransferInterface](#) implementation plugin that transfers data using MPI.

Definition at line 24 of file MPITransfer.hh.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 `artdaq::MPITransfer::MPITransfer (fhicl::ParameterSet pset, TransferInterface::Role role)`

`MPITransfer` Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure <code>MPITransfer</code>
<i>role</i>	Role of this <code>MPITransfer</code> instance (kSend or kReceive)

* `MPITransfer` accepts the following Parameters:
* "synchronous_sends" (Default: true): When false, use `MPI_ISend`, otherwise, use `MPI_SSend`
*

`MPITransfer` also requires all Parameters for configuring a `TransferInterface`

Definition at line 25 of file `MPI_transfer.cc`.

7.41.3 Member Function Documentation

7.41.3.1 `artdaq::TransferInterface::CopyStatus artdaq::MPITransfer::copyFragment (Fragment & frag, size_t timeout_usec = std::numeric_limits<size_t>::max()) [override]`

Copy a Fragment to the destination. Forces asynchronous send.

Parameters

<i>frag</i>	Fragment to copy
<i>timeout_usec</i>	Timeout for send, in microseconds

Returns

`CopyStatus` detailing result of copy

Definition at line 70 of file `MPI_transfer.cc`.

7.41.3.2 `artdaq::TransferInterface::CopyStatus artdaq::MPITransfer::moveFragment (Fragment && frag, size_t timeout_usec = std::numeric_limits<size_t>::max()) [override]`

Move a Fragment to the destination.

Parameters

<i>frag</i>	Fragment to move
<i>timeout_usec</i>	Timeout for send, in microseconds

Returns

`CopyStatus` detailing result of copy

Definition at line 77 of file `MPI_transfer.cc`.

7.41.3.3 `MPITransfer& artdaq::MPITransfer::operator= (const MPITransfer &) [delete]`

Copy Assignment operator is deleted.

Returns

`MPITransfer` reference

7.41.3.4 `int artdaq::MPITransfer::receiveFragmentData (RawDataType * destination, size_t wordCount) [override], [virtual]`

Receive the body of a Fragment to the given destination pointer.

Parameters

<code>destination</code>	Pointer to memory region where Fragment data should be stored
<code>wordCount</code>	Number of words of Fragment data to receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implements `artdaq::TransferInterface`.

Definition at line 200 of file `MPI_transfer.cc`.

7.41.3.5 `int artdaq::MPITransfer::receiveFragmentHeader (detail::RawFragmentHeader & header, size_t receiveTimeout) [override], [virtual]`

Receive a Fragment Header from the transport mechanism.

Parameters

<code>out</code>	<code>header</code>	Received Fragment Header
	<code>receiveTimeout</code>	Timeout for receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implements `artdaq::TransferInterface`.

Definition at line 138 of file `MPI_transfer.cc`.

The documentation for this class was generated from the following files:

- `artdaq/artdaq/TransferPlugins/MPITransfer.hh`
- `artdaq/artdaq/TransferPlugins/MPI_transfer.cc`

7.42 MPRGlobalTestFixture Class Reference

MasterProductRegistry Test Fixture.

Public Types

- `typedef std::map< std::string, art::BranchKey > BKmap_t`
`BKmap_t associates a string with a art::BranchKey.`

Public Member Functions

- `MPRGlobalTestFixture ()`
`MPRGlobalTestFixture Constructor.`
- `art::ProcessConfiguration * fake_single_module_process (std::string const &tag, std::string const &processName, fhicl::ParameterSet const &moduleParams, std::string const &release=art::getReleaseVersion(), std::string const &pass=art::getPassID())`
`Create the ProcessConfiguration for a single module art process.`
- `std::unique_ptr< art::BranchDescription > fake_single_process_branch (std::string const &tag, std::string const &processName, std::string const &productName=std::string())`
`Create a BranchDescription for a process.`
- `void finalize ()`
`Finalizes the ProductRegistry.`

Public Attributes

- `BKmap_t branchKeys_`
`Keys in this test fixture.`
- `std::map< std::string, std::unique_ptr< art::ProcessConfiguration > > processConfigurations_`
`Configurations.`
- `art::MasterProductRegistry productRegistry_`
`MasterProductRegistry instance.`

7.42.1 Detailed Description

MasterProductRegistry Test Fixture.

Definition at line 41 of file shared_memory_reader_t.cc.

7.42.2 Member Function Documentation

7.42.2.1 `art::ProcessConfiguration * MPRGlobalTestFixture::fake_single_module_process (std::string const & tag, std::string const & processName, fhicl::ParameterSet const & moduleParams, std::string const & release = art::getReleaseVersion(), std::string const & pass = art::getPassID())`

Create the ProcessConfiguration for a single module art process.

Parameters

<i>tag</i>	Tag for the ProcessConfiguraiton
<i>processName</i>	Name of the process
<i>moduleParams</i>	ParameterSet for the single module
<i>release</i>	See art::ProcessConfiguration
<i>pass</i>	See art::ProcessConfiguration

Returns

Pointer to created art::ProcessConfiguration object

Definition at line 123 of file shared_memory_reader_t.cc.

7.42.2.2 std::unique_ptr<art::BranchDescription> MPRGlobalTestFixture::fake_single_process_branch (std::string const & *tag*, std::string const & *processName*, std::string const & *productInstanceName* = std::string())

Create a BranchDescription for a process.

Parameters

<i>tag</i>	Tag for the module_process
<i>processName</i>	Name of the process
<i>productInstance-Name</i>	Name of the product

Returns

Pointer to created art::BranchDescription object

Definition at line 141 of file shared_memory_reader_t.cc.

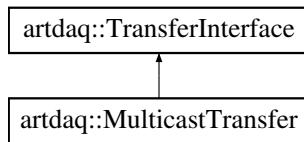
The documentation for this class was generated from the following file:

- artdaq/test/ArtModules/shared_memory_reader_t.cc

7.43 artdaq::MulticastTransfer Class Reference

MulticastTransfer is a [TransferInterface](#) implementation plugin that transfers data using Multicast.

Inheritance diagram for artdaq::MulticastTransfer:



Public Types

- using `byte_t` = artdaq::Fragment::byte_t
Copy Fragment::byte_t into local scope.

Public Member Functions

- virtual `~MulticastTransfer ()=default`
Default destructor.
- `MulticastTransfer (fhicl::ParameterSet const &ps, Role role)`
MulticastTransfer Constructor.
- int `receiveFragment (artdaq::Fragment &fragment, size_t receiveTimeout) override`
Receive a Fragment using Multicast.
- int `receiveFragmentHeader (detail::RawFragmentHeader &header, size_t receiveTimeout) override`
Receive a Fragment Header from the transport mechanism.
- int `receiveFragmentData (RawDataType *destination, size_t wordCount) override`
Receive the body of a Fragment to the given destination pointer.
- `CopyStatus copyFragment (artdaq::Fragment &fragment, size_t send_timeout_usec=std::numeric_limits< size_t >::max()) override`
Copy a Fragment to the destination. Multicast is always unreliable.
- `CopyStatus moveFragment (artdaq::Fragment &&fragment, size_t send_timeout_usec=std::numeric_limits< size_t >::max()) override`
Move a Fragment to the destination. Multicast is always unreliable.

Additional Inherited Members

7.43.1 Detailed Description

`MulticastTransfer` is a `TransferInterface` implementation plugin that transfers data using Multicast.

Definition at line 28 of file `Multicast_transfer.cc`.

7.43.2 Constructor & Destructor Documentation

7.43.2.1 artdaq::MulticastTransfer::MulticastTransfer (`fhicl::ParameterSet const & ps, Role role`)

`MulticastTransfer` Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure <code>MulticastTransfer</code>
<code>role</code>	Role of this <code>MulticastTransfer</code> instance (kSend or kReceive)

```
* MulticastTransfer accepts the following Parameters:
* "subfragment_size" (REQUIRED): Size of the sub-Fragments
* "subfragments_per_send" (REQUIRED): How many sub-Fragments to send in each batch
* "pause_on_copy_usecs" (Default: 0): Pause after sending a batch of sub-Fragments for this many microseconds
* "multicast_port" (REQUIRED): Port number to connect to
* "multicast_address" (REQUIRED): Multicast address to send to/receive from
* "local_address" (REQUIRED): Local origination address for multicast
* "receive_buffer_size" (Default: 0): The UDP receive buffer size. 0 uses automatic size.
*
```

`MulticastTransfer` also requires all Parameters for configuring a `TransferInterface`

Definition at line 156 of file `Multicast_transfer.cc`.

7.43.3 Member Function Documentation

7.43.3.1 **artdaq::TransferInterface::CopyStatus artdaq::MulticastTransfer::copyFragment (artdaq::Fragment & *fragment*, size_t *send_timeout_usec* = std::numeric_limits<size_t>::max()) [override], [virtual]**

Copy a Fragment to the destination. Multicast is always unreliable.

Parameters

<i>fragment</i>	Fragment to copy
<i>send_timeout_usec</i>	Timeout for send, in microseconds

Returns

CopyStatus detailing result of copy

Implements [artdaq::TransferInterface](#).

Definition at line 409 of file Multicast_transfer.cc.

```
7.43.3.2 artdaq::TransferInterface::CopyStatus artdaq::MulticastTransfer::moveFragment( artdaq::Fragment &&
    fragment, size_t send_timeout_usec = std::numeric_limits<size_t>::max() ) [override], [virtual]
```

Move a Fragment to the destination. Multicast is always unreliable.

Parameters

<i>fragment</i>	Fragment to move
<i>send_timeout_usec</i>	Timeout for send, in microseconds

Returns

CopyStatus detailing result of copy

Implements [artdaq::TransferInterface](#).

Definition at line 403 of file Multicast_transfer.cc.

```
7.43.3.3 int artdaq::MulticastTransfer::receiveFragment( artdaq::Fragment & fragment, size_t receiveTimeout ) [override], [virtual]
```

Receive a Fragment using Multicast.

Parameters

<i>out</i>	<i>fragment</i>	Received Fragment
	<i>receiveTimeout</i>	Timeout for receive, in microseconds

Returns

Rank of sender or RECV_TIMEOUT

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 239 of file Multicast_transfer.cc.

```
7.43.3.4 int artdaq::MulticastTransfer::receiveFragmentData( RawDataType * destination, size_t wordCount ) [override], [virtual]
```

Receive the body of a Fragment to the given destination pointer.

Parameters

<i>destination</i>	Pointer to memory region where Fragment data should be stored
<i>wordCount</i>	Number of words of Fragment data to receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implements [artdaq::TransferInterface](#).

Definition at line 390 of file Multicast_transfer.cc.

7.43.3.5 int artdaq::MulticastTransfer::receiveFragmentHeader (detail::RawFragmentHeader & header, size_t receiveTimeout) [override], [virtual]

Receive a Fragment Header from the transport mechanism.

Parameters

<i>out</i>	<i>header</i>	Received Fragment Header
	<i>receiveTimeout</i>	Timeout for receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implements [artdaq::TransferInterface](#).

Definition at line 379 of file Multicast_transfer.cc.

The documentation for this class was generated from the following file:

- artdaq/artdaq/TransferPlugins/Multicast_transfer.cc

7.44 artdaq::NetMonHeader Struct Reference

Header with length information for NetMonTransport messages.

```
#include <artdaq/DAQdata/NetMonHeader.hh>
```

Public Attributes

- `uint64_t data_length`
The length of the message.

7.44.1 Detailed Description

Header with length information for NetMonTransport messages.

Definition at line 14 of file NetMonHeader.hh.

The documentation for this struct was generated from the following file:

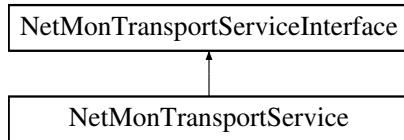
- artdaq/artdaq/DAQdata/NetMonHeader.hh

7.45 NetMonTransportService Class Reference

`NetMonTransportService` extends `NetMonTransportServiceInterface`. It sends events using `DataSenderManager` and receives events from the `GlobalQueue`.

```
#include <artdaq/ArtModules/NetMonTransportService.h>
```

Inheritance diagram for `NetMonTransportService`:



Public Member Functions

- virtual `~NetMonTransportService ()`
`NetMonTransportService` Destructor. Calls `disconnect()`.
- `NetMonTransportService (fhicl::ParameterSet const &pset, art::ActivityRegistry &`
`NetMonTransportService` Constructor.
- void `connect ()` override
Reconnect the `NetMonTransportService`.
- void `disconnect ()` override
Disconnects the `NetMonTransportService`.
- void `listen ()` override
Listen for connections. This method is a No-Op.
- void `sendMessage (uint64_t sequenceld, uint8_t messageType, TBufferFile &msg)` override
Send ROOT data, wrapped in an `artdaq::Fragment` object.
- void `receiveMessage (TBufferFile *&msg)` override
Receive data from the `ConcurrentQueue`.
- void `receiveInitMessage (TBufferFile *&msg)` override
Receive the init message.
- `size_t dataReceiverCount () const`
Get the number of data receivers.

7.45.1 Detailed Description

`NetMonTransportService` extends `NetMonTransportServiceInterface`. It sends events using `DataSenderManager` and receives events from the `GlobalQueue`.

Definition at line 16 of file `NetMonTransportService.h`.

7.45.2 Constructor & Destructor Documentation

7.45.2.1 `NetMonTransportService::NetMonTransportService (fhicl::ParameterSet const & pset, art::ActivityRegistry &)`

`NetMonTransportService` Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure NetMonTransportService and DataSenderManager
-------------	---

* NetMonTransportService accepts the following Parameters
 * "rank" (OPTIONAL): The rank of this application, for use by non-artdaq applications running NetMonTransportService
 *

Definition at line 30 of file NetMonTransportService_service.cc.

7.45.3 Member Function Documentation

7.45.3.1 void NetMonTransportService::connect() [override], [virtual]

Reconnect the [NetMonTransportService](#).

Creates a new instance of DataSenderManager using the stored ParameterSet

Implements [NetMonTransportServiceInterface](#).

Definition at line 50 of file NetMonTransportService_service.cc.

7.45.3.2 size_t NetMonTransportService::dataReceiverCount() const [inline]

Get the number of data receivers.

Returns

The number of data receivers

Definition at line 74 of file NetMonTransportService.h.

7.45.3.3 void NetMonTransportService::disconnect() [override], [virtual]

Disconnects the NetMonTransportService.

Destructs the DataSenderManager

Implements [NetMonTransportServiceInterface](#).

Definition at line 64 of file NetMonTransportService_service.cc.

7.45.3.4 void NetMonTransportService::receiveInitMessage(TBufferFile *& msg) [override], [virtual]

Receive the init message.

Parameters

<i>out</i>	<i>msg</i>	ROOT message data
------------	------------	-------------------

Implements [NetMonTransportServiceInterface](#).

Definition at line 199 of file NetMonTransportService_service.cc.

7.45.3.5 void NetMonTransportService::receiveMessage(TBufferFile *& msg) [override], [virtual]

Receive data from the ConcurrentQueue.

Parameters

out	msg	Received data
-----	-----	---------------

Implements [NetMonTransportServiceInterface](#).

Definition at line 101 of file NetMonTransportService_service.cc.

```
7.45.3.6 void NetMonTransportService::sendMessage ( uint64_t sequenceId, uint8_t messageType, TBufferFile & msg )
[override], [virtual]
```

Send ROOT data, wrapped in an artdaq::Fragment object.

Parameters

sequenceId	The sequence id of the Fragment which will wrap the ROOT data
messageType	The type id of the Fragment which will wrap the ROOT data
msg	The ROOT data to send

Implements [NetMonTransportServiceInterface](#).

Definition at line 71 of file NetMonTransportService_service.cc.

The documentation for this class was generated from the following files:

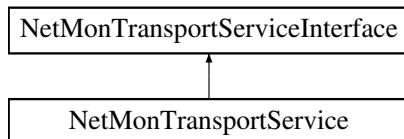
- artdaq/artdaq/ArtModules/NetMonTransportService.h
- artdaq/artdaq/ArtModules/NetMonTransportService_service.cc

7.46 NetMonTransportServiceInterface Class Reference

Interface for NetMonTransportService. This interface is declared to art as part of the required registration of an art Service.

```
#include <artdaq/ArtModules/NetMonTransportServiceInterface.h>
```

Inheritance diagram for NetMonTransportServiceInterface:



Public Member Functions

- virtual [~NetMonTransportServiceInterface](#) ()=default
Default virtual destructor.
- virtual void [connect](#) ()=0
Connect the NetMonTransportService.
- virtual void [disconnect](#) ()=0
Disconnect the NetMonTransportService.
- virtual void [listen](#) ()=0
Listen for new connections.

- virtual void [sendMessage](#) (uint64_t sequenceId, uint8_t messageType, TBufferFile &msg)=0
Send a message.
- virtual void [receiveMessage](#) (TBufferFile *&msg)=0
Receive a message.
- virtual void [receiveInitMessage](#) (TBufferFile *&msg)=0
Receive the init message.

7.46.1 Detailed Description

Interface for NetMonTransportService. This interface is declared to act as part of the required registration of an art Service.

Definition at line 11 of file NetMonTransportServiceInterface.h.

7.46.2 Member Function Documentation

7.46.2.1 virtual void NetMonTransportServiceInterface::connect() [pure virtual]

Connect the [NetMonTransportService](#).

This is a pure virtual function, derived classes must reimplement it

Implemented in [NetMonTransportService](#).

7.46.2.2 virtual void NetMonTransportServiceInterface::disconnect() [pure virtual]

Disconnect the [NetMonTransportService](#).

This is a pure virtual function, derived classes must reimplement it

Implemented in [NetMonTransportService](#).

7.46.2.3 virtual void NetMonTransportServiceInterface::listen() [pure virtual]

Listen for new connections.

This is a pure virtual function, derived classes must reimplement it

Implemented in [NetMonTransportService](#).

7.46.2.4 virtual void NetMonTransportServiceInterface::receiveInitMessage(TBufferFile *& msg) [pure virtual]

Receive the init message.

Parameters

out	msg	ROOT message data
-----	-----	-------------------

Implemented in [NetMonTransportService](#).

7.46.2.5 virtual void NetMonTransportServiceInterface::receiveMessage(TBufferFile *& msg) [pure virtual]

Receive a message.

Parameters

<code>out</code>	<code>msg</code>	ROOT message data
------------------	------------------	-------------------

Implemented in [NetMonTransportService](#).

7.46.2.6 `virtual void NetMonTransportServiceInterface::sendMessage (uint64_t sequenceId, uint8_t messageType, TBufferFile & msg) [pure virtual]`

Send a message.

Parameters

<code>sequenceId</code>	Sequence ID of Fragment wrapping the message
<code>messageType</code>	Fragment type of Fragment wrapping the message
<code>msg</code>	ROOT data to send

Implemented in [NetMonTransportService](#).

The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/NetMonTransportServiceInterface.h

7.47 art::NetMonWrapper Class Reference

This class wraps [NetMonTransportService](#) so that it can act as an [ArtdaqInput](#) template class.

```
#include <artdaq/ArtModules/NetMonWrapper.hh>
```

Public Member Functions

- [NetMonWrapper \(const fhicl::ParameterSet &\)](#)
NetMonWrapper Constructor.
- [~NetMonWrapper \(\)](#)
NetMonWrapper Destructor.
- [void receiveMessage \(std::unique_ptr< TBufferFile > &msg\)](#)
Receive a message from the NetMonTransportService.
- [void receiveInitMessage \(std::unique_ptr< TBufferFile > &msg\)](#)
Receive an init message from the NetMonTransportService.

7.47.1 Detailed Description

This class wraps [NetMonTransportService](#) so that it can act as an [ArtdaqInput](#) template class.

JCF, May-27-2016

This class is written with functionality such that it satisfies the requirements needed to be a template in the [ArtdaqInput](#) class

Definition at line 26 of file NetMonWrapper.hh.

7.47.2 Constructor & Destructor Documentation

7.47.2.1 `art::NetMonWrapper::NetMonWrapper (const fhicl::ParameterSet &) [inline]`

[NetMonWrapper](#) Constructor.

JCF, May-31-2016

Parameter set constructor argument is unused for now, but needed for this class to implement the interface the [Artdaq-Input](#) templatized input source expects

Definition at line 39 of file NetMonWrapper.hh.

7.47.3 Member Function Documentation

7.47.3.1 `void art::NetMonWrapper::receiveInitMessage (std::unique_ptr< TBufferFile > & msg)`

Receive an init message from the [NetMonTransportService](#).

Parameters

<code>out</code>	<code>msg</code>	A pointer to the received message
------------------	------------------	-----------------------------------

Definition at line 17 of file NetMonWrapper.cc.

7.47.3.2 `void art::NetMonWrapper::receiveMessage (std::unique_ptr< TBufferFile > & msg)`

Receive a message from the [NetMonTransportService](#).

Parameters

<code>out</code>	<code>msg</code>	A pointer to the received message
------------------	------------------	-----------------------------------

Definition at line 5 of file NetMonWrapper.cc.

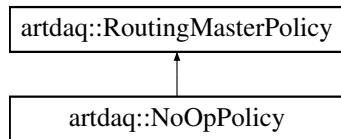
The documentation for this class was generated from the following files:

- artdaq/artdaq/ArtModules/NetMonWrapper.hh
- artdaq/artdaq/ArtModules/NetMonWrapper.cc

7.48 artdaq::NoOpPolicy Class Reference

A [RoutingMasterPolicy](#) which simply assigns Sequence IDs to tokens in the order they were received.

Inheritance diagram for artdaq::NoOpPolicy:



Public Member Functions

- [NoOpPolicy](#) (fhicl::ParameterSet ps)
NoOpPolicy Constructor.
- virtual ~[NoOpPolicy](#) ()=default
Default virtual Destructor.
- [detail::RoutingPacket GetCurrentTable](#) () override
Using the tokens received so far, create a Routing Table.

Additional Inherited Members

7.48.1 Detailed Description

A [RoutingMasterPolicy](#) which simply assigns Sequence IDs to tokens in the order they were received.

Definition at line 10 of file NoOp_policy.cc.

7.48.2 Constructor & Destructor Documentation

7.48.2.1 artdaq::NoOpPolicy::NoOpPolicy (fhicl::ParameterSet ps) [inline], [explicit]

[NoOpPolicy](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure the NoOpPolicy
-----------	---

[NoOpPolicy](#) takes no additional Parameters at this time

Definition at line 19 of file NoOp_policy.cc.

7.48.3 Member Function Documentation

7.48.3.1 detail::RoutingPacket artdaq::NoOpPolicy::GetCurrentTable () [override], [virtual]

Using the tokens received so far, create a Routing Table.

Returns

A [detail::RoutingPacket](#) containing the Routing Table

Implements [artdaq::RoutingMasterPolicy](#).

Definition at line 33 of file NoOp_policy.cc.

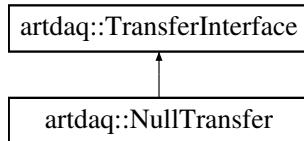
The documentation for this class was generated from the following file:

- artdaq/artdaq/Application/Routing/NoOp_policy.cc

7.49 artdaq::NullTransfer Class Reference

[NullTransfer](#) does not send or receive data, but acts as if it did.

Inheritance diagram for artdaq::NullTransfer:



Public Member Functions

- `NullTransfer (const fhicl::ParameterSet &pset, Role role)`
NullTransfer constructor.
- `virtual ~NullTransfer ()=default`
NullTransfer default Destructor.
- `int receiveFragment (artdaq::Fragment &, size_t) override`
Pretend to receive a Fragment.
- `int receiveFragmentHeader (detail::RawFragmentHeader &, size_t) override`
Pretend to receive a Fragment Header.
- `int receiveFragmentData (RawDataType *, size_t) override`
Pretend to receive Fragment Data.
- `CopyStatus copyFragment (artdaq::Fragment &, size_t) override`
Pretend to copy a Fragment to a destination.
- `CopyStatus moveFragment (artdaq::Fragment &&, size_t) override`
Pretend to move a Fragment to a destination.

Additional Inherited Members

7.49.1 Detailed Description

`NullTransfer` does not send or receive data, but acts as if it did.

Definition at line 8 of file `Null_transfer.cc`.

7.49.2 Constructor & Destructor Documentation

7.49.2.1 artdaq::NullTransfer::NullTransfer (const fhicl::ParameterSet & pset, Role role)

`NullTransfer` constructor.

Parameters

<code>pset</code>	ParameterSet used to configure <code>TransferInterface</code>
<code>role</code>	Role of this <code>NullTransfer</code> instance (kSend or kReceive)

`NullTransfer` only requires the Parameters for configuring a `TransferInterface`

Definition at line 81 of file `Null_transfer.cc`.

7.49.3 Member Function Documentation

7.49.3.1 **CopyStatus artdaq::NullTransfer::copyFragment(artdaq::Fragment &, size_t) [inline], [override], [virtual]**

Pretend to copy a Fragment to a destination.

Returns

[CopyStatus::kSuccess](#) (No-Op)

Implements [artdaq::TransferInterface](#).

Definition at line 65 of file Null_transfer.cc.

7.49.3.2 **CopyStatus artdaq::NullTransfer::moveFragment(artdaq::Fragment &&, size_t) [inline], [override], [virtual]**

Pretend to move a Fragment to a destination.

Returns

[CopyStatus::kSuccess](#) (No-Op)

Implements [artdaq::TransferInterface](#).

Definition at line 74 of file Null_transfer.cc.

7.49.3.3 **int artdaq::NullTransfer::receiveFragment(artdaq::Fragment &, size_t) [inline], [override], [virtual]**

Pretend to receive a Fragment.

Returns

Source Rank (Success code)

WARNING: This function may create unintended side-effects. [NullTransfer](#) should only really be used in [Role::kSend](#)!

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 32 of file Null_transfer.cc.

7.49.3.4 **int artdaq::NullTransfer::receiveFragmentData(RawDataType * , size_t) [inline], [override], [virtual]**

Pretend to receive Fragment Data.

Returns

Source Rank (Success code)

WARNING: This function may create unintended side-effects. [NullTransfer](#) should only really be used in [Role::kSend](#)!

Implements [artdaq::TransferInterface](#).

Definition at line 56 of file Null_transfer.cc.

7.49.3.5 int artdaq::NullTransfer::receiveFragmentHeader (detail::RawFragmentHeader & , size_t) [inline],
 [override], [virtual]

Pretend to receive a Fragment Header.

Returns

Source Rank (Success code)

WARNING: This function may create unintended side-effects. [NullTransfer](#) should only really be used in [Role::kSend!](#)

Implements [artdaq::TransferInterface](#).

Definition at line 44 of file Null_transfer.cc.

The documentation for this class was generated from the following file:

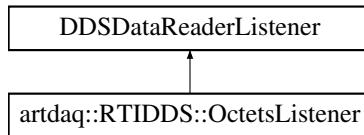
- artdaq/artdaq/TransferPlugins/Null_transfer.cc

7.50 artdaq::RTIDDS::OctetsListener Class Reference

A class that reads data from DDS.

```
#include <artdaq/RTIDDS/RTIDDS.hh>
```

Inheritance diagram for artdaq::RTIDDS::OctetsListener:



Public Member Functions

- void [on_data_available](#) (DDSDataReader *reader)
Action to perform when data is available.
- bool [receiveFragmentFromDDS](#) (artdaq::Fragment &fragment, const size_t receiveTimeout)
Receive a Fragment from DDS.

7.50.1 Detailed Description

A class that reads data from DDS.

Definition at line 70 of file RTIDDS.hh.

7.50.2 Member Function Documentation

7.50.2.1 void artdaq::RTIDDS::OctetsListener::on_data_available (DDSDataReader * reader)

Action to perform when data is available.

Parameters

<code>reader</code>	Reader reference to read data from
---------------------	------------------------------------

Definition at line 164 of file RTIDDS.cc.

7.50.2.2 `bool artdaq::RTIDDS::OctetsListener::receiveFragmentFromDDS (artdaq::Fragment & fragment, const size_t receiveTimeout)`

Receive a Fragment from DDS.

Parameters

<code>out</code>	<code>fragment</code>	Received Fragment
	<code>receiveTimeout</code>	Timeout for receive operation

Returns

Whether the receive succeeded in receiveTimeout

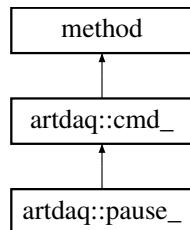
Definition at line 216 of file RTIDDS.cc.

The documentation for this class was generated from the following files:

- artdaq/artdaq/RTIDDS/RTIDDS.hh
- artdaq/artdaq/RTIDDS/RTIDDS.cc

7.51 artdaq::pause_ Class Reference

Inheritance diagram for artdaq::pause_:



Public Member Functions

- `pause_ (xmlrpc_commander &c)`

Static Public Attributes

- static const uint64_t `defaultTimeout` = 45
- static const uint64_t `defaultTimestamp` = std::numeric_limits<const uint64_t>::max()

Additional Inherited Members

7.51.1 Detailed Description

[pause_](#) Command class

Definition at line 473 of file `xmlrpc_commander.cc`.

7.51.2 Constructor & Destructor Documentation

7.51.2.1 artdaq::pause_::pause_(`xmlrpc_commander & c`) [inline]

[pause_](#) Constructor \

Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send transition commands to
----------------	--

Definition at line 473 of file `xmlrpc_commander.cc`.

7.51.3 Member Data Documentation

7.51.3.1 const uint64_t artdaq::pause_::defaultTimeout = 45 [static]

Default timeout for command

Definition at line 473 of file `xmlrpc_commander.cc`.

7.51.3.2 const uint64_t artdaq::pause_::defaultTimestamp = std::numeric_limits<const uint64_t>::max() [static]

Default timestamp for Command

Definition at line 473 of file `xmlrpc_commander.cc`.

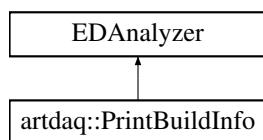
The documentation for this class was generated from the following file:

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

7.52 artdaq::PrintBuildInfo Class Reference

An art::EDAnalyzer which prints any [artdaq::BuildInfo](#) objects stored in the run.

Inheritance diagram for artdaq::PrintBuildInfo:



Public Member Functions

- `PrintBuildInfo (fhicl::ParameterSet const &p)`
PrintBuildInfo Constructor.
- `virtual ~PrintBuildInfo ()=default`
Default virtual Destructor.
- `void analyze (art::Event const &)` override
Called for each event. Required overload for art::EDAnalyzer, No-Op here.
- `void beginRun (art::Run const &run)` override
Perform actions at the beginning of the run.

7.52.1 Detailed Description

An art::EDAnalyzer which prints any `artdaq::BuildInfo` objects stored in the run.

Definition at line 32 of file `PrintBuildInfo_module.cc`.

7.52.2 Constructor & Destructor Documentation

7.52.2.1 `artdaq::PrintBuildInfo::PrintBuildInfo (fhicl::ParameterSet const & p)` [explicit]

`PrintBuildInfo` Constructor.

Parameters

<code>p</code>	ParameterSet used to configure <code>PrintBuildInfo</code>
----------------	--

* `PrintBuildInfo` accepts the following Parameters:
* "buildinfo_module_label" (REQUIRED): The module label for the `BuildInfo` objects
* "buildinfo_instance_label" (REQUIRED): The instance label for the `BuildInfo` objects
*

These parameters should match those given to the `BuildInfo` module

Definition at line 75 of file `PrintBuildInfo_module.cc`.

7.52.3 Member Function Documentation

7.52.3.1 `void artdaq::PrintBuildInfo::beginRun (art::Run const & run)` [override]

Perform actions at the beginning of the run.

Parameters

<code>run</code>	art::Run object
------------------	-----------------

This function pretty-prints the `BuildInfo` information form the run object with the configured module label and instance label.

Definition at line 83 of file `PrintBuildInfo_module.cc`.

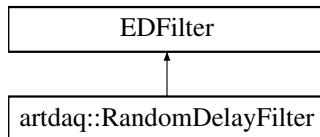
The documentation for this class was generated from the following file:

- `artdaq/artdaq/ArtModules/PrintBuildInfo_module.cc`

7.53 artdaq::RandomDelayFilter Class Reference

A filter which delays for a random amount of time, then drops a random fraction of events. Used to simulate the delays and efficiency of real filters.

Inheritance diagram for artdaq::RandomDelayFilter:



Public Member Functions

- [RandomDelayFilter \(fhicl::ParameterSet const &p\)](#)
RandomDelayFilter Constructor.
- [RandomDelayFilter \(RandomDelayFilter const &\)=delete](#)
Copy Constructor is deleted.
- [RandomDelayFilter \(RandomDelayFilter &&\)=delete](#)
Move Constructor is deleted.
- [RandomDelayFilter & operator= \(RandomDelayFilter const &\)=delete](#)
Copy Assignment operator is deleted.
- [RandomDelayFilter & operator= \(RandomDelayFilter &&\)=delete](#)
Move Assignment operator is deleted.
- [bool filter \(art::Event &e\) override](#)
Filter is a required override of art::EDFilter, and is called for each event.
- [void reconfigure \(fhicl::ParameterSet const &p\) override](#)
Reconfigure the RandomDelayFilter.

7.53.1 Detailed Description

A filter which delays for a random amount of time, then drops a random fraction of events. Used to simulate the delays and efficiency of real filters.

Multiple RandomDelayFilters in series can simulate the effect of multiple layers of filtering

Definition at line 37 of file RandomDelayFilter_module.cc.

7.53.2 Constructor & Destructor Documentation

7.53.2.1 artdaq::RandomDelayFilter::RandomDelayFilter (fhicl::ParameterSet const & p) [explicit]

[RandomDelayFilter](#) Constructor.

Parameters

<i>p</i>	ParameterSet used to configure RandomDelayFilter
----------	--

[RandomDelayFilter](#) accepts the following Parameters:

- "minimum_delay_ms" (Default: 0): The minimum amount of time to delay, in ms
- "maximum_delay_ms" (Default: 1000): The maximum amount of time to delay, in ms
- "mean_delay_ms" (Default: 500): If using a normal distribution for delay times, the mean of the distribution, in ms
- "sigma_delay_ms" (Default: 100): If using a normal distribution for delay times, the sigma of the distribution, in ms
- "pass_filter_percentage" (Default: 100): The fraction of events which will pass the filter
- "cpu_load_ratio" (Default: 1.0): The fraction of the delay time which should be active (spinning) versus passive (sleeping)
- "use_normal_distribution" (Default: false): Use a normal distribution for delay times, versus a uniform one
- "random_seed" (Default: 271828): The seed for the distribution

Definition at line 124 of file RandomDelayFilter_module.cc.

7.53.3 Member Function Documentation

7.53.3.1 bool artdaq::RandomDelayFilter::filter (art::Event & e) [override]

Filter is a required override of art::EDFilter, and is called for each event.

Parameters

<i>e</i>	The art::Event to filter
----------	--------------------------

Returns

Whether the event passes the filter

This function is where [RandomDelayFilter](#) performs its work, using the delay distribution to pick a delay time, spinning and/or sleeping for that amount of time, then picking an integer from the pass distribution to determine if the event should pass or not.

Definition at line 150 of file RandomDelayFilter_module.cc.

7.53.3.2 RandomDelayFilter& artdaq::RandomDelayFilter::operator= (RandomDelayFilter const &) [delete]

Copy Assignment operator is deleted.

Returns

[RandomDelayFilter](#) copy

7.53.3.3 RandomDelayFilter& artdaq::RandomDelayFilter::operator= (RandomDelayFilter &&) [delete]

Move Assignment operator is deleted.

Returns

[RandomDelayFilter](#) instance

7.53.3.4 void artdaq::RandomDelayFilter::reconfigure (fhicl::ParameterSet const & p) [override]

Reconfigure the [RandomDelayFilter](#).

Parameters

<i>p</i>	ParameterSet used to configure the RandomDelayFilter
----------	--

[RandomDelayFilter](#) accepts the following Parameters: "minimum_delay_ms" (Default: 0): The minimum amount of time to delay, in ms "maximum_delay_ms" (Default: 1000): The maximum amount of time to delay, in ms "mean_delay_ms" (Default: 500): If using a normal distribution for delay times, the mean of the distribution, in ms "sigma_delay_ms" (Default: 100): If using a normal distribution for delay times, the sigma of the distribution, in ms "pass_filter_percentage" (Default: 100): The fraction of events which will pass the filter "cpu_load_ratio" (Default: 1.0): The fraction of the delay time which should be active (spinning) versus passive (sleeping) "use_normal_distribution" (Default: false): Use a normal distribution for delay times, versus a uniform one "random_seed" (Default: 271828): The seed for the distribution

Definition at line 167 of file RandomDelayFilter_module.cc.

The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/RandomDelayFilter_module.cc

7.54 daqrate.Re Class Reference

General Regular Expression class that allows for: xx = Re(re) ...

Public Member Functions

- def [__init__](#)
Create an instance of the regular expression.
- def [search](#)
Find matches for the regular expression.

Public Attributes

- [compiled](#)
The compiled version of the regex.
- [match_obj](#)
Match object representing all matches for the regex.

7.54.1 Detailed Description

General Regular Expression class that allows for: xx = Re(re) ...

if xx.search(line): yy = xx.match_obj.group(x)

Definition at line 47 of file daqrate.py.

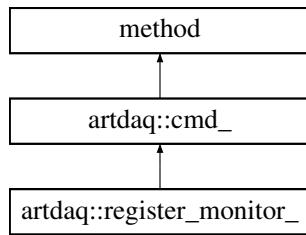
The documentation for this class was generated from the following file:

- artdaq/proto/daqrate.py

7.55 artdaq::register_monitor_ Class Reference

[register_monitor_](#) Command class

Inheritance diagram for artdaq::register_monitor_:



Public Member Functions

- [register_monitor_\(xmlrpc_commander &c\)](#)
register_monitor_ Constructor

Additional Inherited Members

7.55.1 Detailed Description

[register_monitor_ Command class](#)

Definition at line 601 of file `xmlrpc_commander.cc`.

7.55.2 Constructor & Destructor Documentation

7.55.2.1 artdaq::register_monitor_::register_monitor_(xmlrpc_commander & c) [inline]

[register_monitor_ Constructor](#)

Parameters

<code>c</code>	xmlrpc_commander to send transition commands to
----------------	---

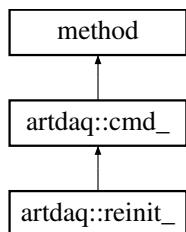
Definition at line 608 of file `xmlrpc_commander.cc`.

The documentation for this class was generated from the following file:

- artdaq/artdaq/ExternalComms/xmlrpc_commander.cc

7.56 artdaq::reinit_ Class Reference

Inheritance diagram for artdaq::reinit_:



Public Member Functions

- [reinit_\(xmlrpc_commander &c\)](#)

Static Public Attributes

- static const uint64_t [defaultTimeout](#) = 45
- static const uint64_t [defaultTimestamp](#) = std::numeric_limits<const uint64_t>::max()

Additional Inherited Members

7.56.1 Detailed Description

Command class representing an init transition

Definition at line 393 of file `xmlrpc_commander.cc`.

7.56.2 Constructor & Destructor Documentation

7.56.2.1 artdaq::reinit_::reinit_(xmlrpc_commander & c) [inline], [explicit]

Command class Constructor *

Parameters

<code>c</code>	xmlrpc_commander to send parsed command to \
----------------	--

Definition at line 393 of file `xmlrpc_commander.cc`.

7.56.3 Member Data Documentation

7.56.3.1 const uint64_t artdaq::reinit_::defaultTimeout = 45 [static]

Default timeout for command

Definition at line 393 of file `xmlrpc_commander.cc`.

7.56.3.2 const uint64_t artdaq::reinit_::defaultTimestamp = std::numeric_limits<const uint64_t>::max() [static]

Default timestamp for Command

Definition at line 393 of file `xmlrpc_commander.cc`.

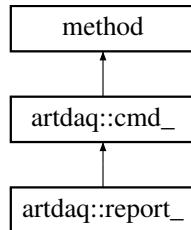
The documentation for this class was generated from the following file:

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

7.57 artdaq::report_ Class Reference

[report_](#) Command class

Inheritance diagram for artdaq::report_:



Public Member Functions

- `report_(xmlrpc_commander &c)`
report_ Constructor

Additional Inherited Members

7.57.1 Detailed Description

`report_` Command class

Definition at line 535 of file `xmlrpc_commander.cc`.

7.57.2 Constructor & Destructor Documentation

7.57.2.1 artdaq::report_::report_(xmlrpc_commander & c) [inline]

`report_` Constructor

Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send transition commands to
----------------	--

Definition at line 542 of file `xmlrpc_commander.cc`.

The documentation for this class was generated from the following file:

- artdaq/artdaq/ExternalComms/xmlrpc_commander.cc

7.58 artdaq::detail::RequestHeader Struct Reference

Header of a `RequestMessage`. Contains magic bytes for validation and a count of expected RequestPackets.

```
#include <artdaq/DAQrate/detail/RequestMessage.hh>
```

Public Member Functions

- `RequestHeader()`
Default Constructor.
- `bool isValid() const`
Check the magic bytes of the packet.

Public Attributes

- `uint32_t header`
- `uint32_t packet_count`

The number of RequestPackets in this Request message.
- `RequestMessageMode mode`

Communicates additional information to the Request receiver.

7.58.1 Detailed Description

Header of a [RequestMessage](#). Contains magic bytes for validation and a count of expected RequestPackets.

Definition at line 87 of file RequestMessage.hh.

7.58.2 Member Function Documentation

7.58.2.1 `bool artdaq::detail::RequestHeader::isValid() const [inline]`

Check the magic bytes of the packet.

Returns

Whether the correct magic bytes were found

Definition at line 106 of file RequestMessage.hh.

7.58.3 Member Data Documentation

7.58.3.1 `uint32_t artdaq::detail::RequestHeader::header`

The magic bytes for the request header

Definition at line 90 of file RequestMessage.hh.

The documentation for this struct was generated from the following file:

- artdaq/artdaq/DAQrate/detail/RequestMessage.hh

7.59 artdaq::detail::RequestMessage Class Reference

A [RequestMessage](#) consists of a [RequestHeader](#) and zero or more RequestPackets. They will usually be sent in two calls to `send()`

```
#include <artdaq/DAQrate/detail/RequestMessage.hh>
```

Public Member Functions

- `RequestMessage()`

Default Constructor.
- `RequestHeader * header()`

Get a pointer to the [RequestHeader](#), filling in the current size of the message.

- [RequestPacket * buffer \(\)](#)

Get a pointer to the first [RequestPacket](#) in contiguous storage.

- [size_t size \(\) const](#)

Get the number of RequestPackets in the [RequestMessage](#).

- [void addRequest \(const Fragment::sequence_id_t &seq, const Fragment::timestamp_t &time\)](#)

Add a request for a sequence ID and timestamp combination.

7.59.1 Detailed Description

A [RequestMessage](#) consists of a [RequestHeader](#) and zero or more RequestPackets. They will usually be sent in two calls to send()

Definition at line 112 of file RequestMessage.hh.

7.59.2 Member Function Documentation

7.59.2.1 void artdaq::detail::RequestMessage::addRequest (const Fragment::sequence_id_t & seq, const Fragment::timestamp_t & time) [inline]

Add a request for a sequence ID and timestamp combination.

Parameters

<i>seq</i>	Sequence ID to request
<i>time</i>	Timestamp of request

Definition at line 148 of file RequestMessage.hh.

7.59.2.2 RequestPacket* artdaq::detail::RequestMessage::buffer () [inline]

Get a pointer to the first [RequestPacket](#) in contiguous storage.

Returns

Pointer to the first Request Packet

Definition at line 136 of file RequestMessage.hh.

7.59.2.3 RequestHeader* artdaq::detail::RequestMessage::header () [inline]

Get a pointer to the [RequestHeader](#), filling in the current size of the message.

Returns

A pointer to the [RequestHeader](#)

Definition at line 126 of file RequestMessage.hh.

7.59.2.4 `size_t artdaq::detail::RequestMessage::size() const [inline]`

Get the number of RequestPackets in the [RequestMessage](#).

Returns

The number of RequestPackets in the [RequestMessage](#)

Definition at line 141 of file RequestMessage.hh.

The documentation for this class was generated from the following file:

- artdaq/artdaq/DAQrate/detail/RequestMessage.hh

7.60 artdaq::detail::RequestPacket Struct Reference

The [RequestPacket](#) contains information about a single data request.

```
#include <artdaq/DAQrate/detail/RequestMessage.hh>
```

Public Member Functions

- [RequestPacket \(\)](#)
Default Constructor.
- [RequestPacket \(const Fragment::sequence_id_t &seq, const Fragment::timestamp_t &ts\)](#)
Create a [RequestPacket](#) using the given sequence ID and timestamp.
- bool [isValid \(\) const](#)
Check the magic bytes of the packet.

Public Attributes

- `uint32_t header`
- `Fragment::sequence_id_t sequence_id`
The sequence ID that responses to this request should use.
- `Fragment::timestamp_t timestamp`
The timestamp of the request.

7.60.1 Detailed Description

The [RequestPacket](#) contains information about a single data request.

Definition at line 49 of file RequestMessage.hh.

7.60.2 Constructor & Destructor Documentation

7.60.2.1 `artdaq::detail::RequestPacket::RequestPacket (const Fragment::sequence_id_t & seq, const Fragment::timestamp_t & ts) [inline]`

Create a [RequestPacket](#) using the given sequence ID and timestamp.

Parameters

<code>seq</code>	Sequence ID of RequestPacket
<code>ts</code>	Timestamp of RequestPAcet

Definition at line 71 of file RequestMessage.hh.

7.60.3 Member Function Documentation

7.60.3.1 `bool artdaq::detail::RequestPacket::isValid() const [inline]`

Check the magic bytes of the packet.

Returns

Whether the correct magic bytes were found

Definition at line 81 of file RequestMessage.hh.

7.60.4 Member Data Documentation

7.60.4.1 `uint32_t artdaq::detail::RequestPacket::header`

The magic bytes for the request packet

Definition at line 53 of file RequestMessage.hh.

The documentation for this struct was generated from the following file:

- artdaq/artdaq/DAQrate/detail/RequestMessage.hh

7.61 artdaq::RequestSender Class Reference

The [RequestSender](#) contains methods used to send data requests and Routing tokens.

```
#include <artdaq/DAQrate/RequestSender.hh>
```

Public Member Functions

- [RequestSender \(\)=delete](#)
Default Constructor is deleted.
- [RequestSender \(RequestSender const &\)=delete](#)
Copy Constructor is deleted.
- [RequestSender & operator= \(RequestSender const &\)=delete](#)
Copy Assignment operator is deleted.
- [RequestSender \(const fhicl::ParameterSet &pset\)](#)
RequestSender Constructor.
- [virtual ~RequestSender \(\)](#)
RequestSender Destructor.
- [void SetRequestMethod \(detail::RequestMessageMode mode\)](#)

- `detail::RequestMessageMode GetRequestMethod () const`
Get the mode for RequestMessages.
- `void SendRequest (bool endOfRunOnly=false)`
Send a request message containing all current requests.
- `void AddRequest (Fragment::sequence_id_t seqID, Fragment::timestamp_t timestamp)`
Add a request to the request list.
- `void RemoveRequest (Fragment::sequence_id_t seqID)`
Remove a request from the request list.
- `void SendRoutingToken (int nSlots)`
Send a RoutingToken message indicating that slots are available.

7.61.1 Detailed Description

The [RequestSender](#) contains methods used to send data requests and Routing tokens.

Definition at line 27 of file RequestSender.hh.

7.61.2 Constructor & Destructor Documentation

7.61.2.1 artdaq::RequestSender::RequestSender (const fhi1::ParameterSet & pset)

[RequestSender](#) Constructor.

Parameters

<code>pset</code>	ParameterSet used to configured RequestSender
-------------------	---

```
* RequestSender accepts the following Parameters:
* "send_requests" (Default: false): Whether to send DataRequests when new sequence IDs are seen
* "request_port" (Default: 3001): Port to send DataRequests on
* "request_delay_ms" (Default: 10): How long to wait before sending new DataRequests
* "request_shutdown_timeout_us" (Default: 0.1s): How long to wait for pending requests to be sent at shutdown
* "output_address" (Default: "localhost"): Use this hostname for multicast output (to assign to the proper NIC)
* "request_address" (Default: "227.128.12.26"): Multicast address to send DataRequests to
* "routing_token_config" (Default: Empty table): FHiCL table containing RoutingToken configuration
* "use_routing_master" (Default: false): Whether to send tokens to a RoutingMaster
* "routing_token_port" (Default: 35555): Port to send tokens on
* "routing_master_hostname" (Default: "localhost"): Hostname or IP of RoutingMaster
*
```

Definition at line 21 of file RequestSender.cc.

7.61.3 Member Function Documentation

7.61.3.1 void artdaq::RequestSender::AddRequest (Fragment::sequence_id_t seqID, Fragment::timestamp_t timestamp)

Add a request to the request list.

Parameters

<i>seqID</i>	Sequence ID for request
<i>timestamp</i>	Timestamp to request

Definition at line 227 of file RequestSender.cc.

7.61.3.2 detail::RequestMessageMode artdaq::RequestSender::GetRequestMethod () const [inline]

Get the mode for RequestMessages.

Returns

Current RequestMessageMode of the [RequestSender](#)

Definition at line 81 of file RequestSender.hh.

7.61.3.3 RequestSender& artdaq::RequestSender::operator= (RequestSender const &) [delete]

Copy Assignment operator is deleted.

Returns

[RequestSender](#) copy

7.61.3.4 void artdaq::RequestSender::RemoveRequest (Fragment::sequence_id_t seqID)

Remove a request from the request list.

Parameters

<i>seqID</i>	Sequence ID of request
--------------	------------------------

Definition at line 236 of file RequestSender.cc.

7.61.3.5 void artdaq::RequestSender::SendRequest (bool endOfRunOnly = false)

Send a request message containing all current requests.

Parameters

<i>endOfRunOnly</i>	Whether the request should only be sent in EndOfRun RequestMessageMode (default: false)
---------------------	---

Definition at line 218 of file RequestSender.cc.

7.61.3.6 void artdaq::RequestSender::SendRoutingToken (int nSlots)

Send a RoutingToken message indicating that slots are available.

Parameters

<code>nSlots</code>	Number of slots available
---------------------	---------------------------

Definition at line 210 of file RequestSender.cc.

7.61.3.7 void artdaq::RequestSender::SetRequestMethod (`detail::RequestMessageMode mode`)

Set the mode for RequestMessages. Used to indicate when [RequestSender](#) should enter "EndOfRun" mode.

Parameters

<code>mode</code>	Mode to set
-------------------	-------------

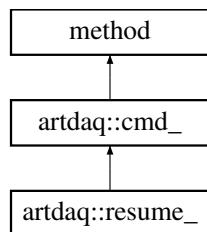
Definition at line 67 of file RequestSender.cc.

The documentation for this class was generated from the following files:

- artdaq/artdaq/DAQrate/RequestSender.hh
- artdaq/artdaq/DAQrate/RequestSender.cc

7.62 artdaq::resume_ Class Reference

Inheritance diagram for artdaq::resume_:



Public Member Functions

- [resume_ \(xmlrpc_commander &c\)](#)

Static Public Attributes

- static const uint64_t `defaultTimeout` = 45
- static const uint64_t `defaultTimestamp` = std::numeric_limits<const uint64_t>::max()

Additional Inherited Members

7.62.1 Detailed Description

[resume_](#) Command class

Definition at line 475 of file xmlrpc_commander.cc.

7.62.2 Constructor & Destructor Documentation

7.62.2.1 `artdaq::resume_::resume_(xmlrpc_commander & c) [inline]`

[resume_ Constructor \](#)

Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send transition commands to
----------------	--

Definition at line 475 of file `xmlrpc_commander.cc`.

7.62.3 Member Data Documentation

7.62.3.1 `const uint64_t artdaq::resume_::defaultTimeout = 45 [static]`

Default timeout for command

Definition at line 475 of file `xmlrpc_commander.cc`.

7.62.3.2 `const uint64_t artdaq::resume_::defaultTimestamp = std::numeric_limits<const uint64_t>::max() [static]`

Default timestamp for Command

Definition at line 475 of file `xmlrpc_commander.cc`.

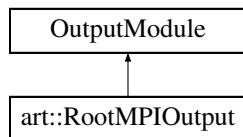
The documentation for this class was generated from the following file:

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

7.63 art::RootMPIOutput Class Reference

An `art::OutputModule` which sends events using `DataSenderManager`. This module is designed for transporting Fragment-wrapped `art::Events` after they have been read into `art`, for example between the `EventBuilder` and the `Aggregator`.

Inheritance diagram for `art::RootMPIOutput`:



Public Member Functions

- `RootMPIOutput (fhicl::ParameterSet const &ps)`

RootMPIOutput Constructor.

- `~RootMPIOutput ()`

RootMPIOutput Destructor.

7.63.1 Detailed Description

An art::OutputModule which sends events using DataSenderManager. This module is designed for transporting Fragment-wrapped art::Events after they have been read into art, for example between the EventBuilder and the Aggregator.

Definition at line 70 of file RootMPIOutput_module.cc.

7.63.2 Constructor & Destructor Documentation

7.63.2.1 art::RootMPIOutput::RootMPIOutput (fhicl::ParameterSet const & ps) [explicit]

[RootMPIOutput](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure RootMPIOutput
-----------	--

[RootMPIOutput](#) accepts no Parameters beyond those which art::OutputModule takes. See the art::OutputModule documentation for more details on those Parameters.

Definition at line 112 of file RootMPIOutput_module.cc.

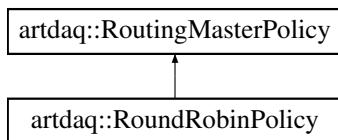
The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/RootMPIOutput_module.cc

7.64 artdaq::RoundRobinPolicy Class Reference

A [RoutingMasterPolicy](#) which evenly distributes Sequence IDs to all receivers. If an uneven number of tokens have been received, extra tokens are stored for the next table update.

Inheritance diagram for artdaq::RoundRobinPolicy:



Public Member Functions

- [RoundRobinPolicy](#) (fhicl::ParameterSet ps)
RoundRobinPolicy Constructor.
- virtual ~[RoundRobinPolicy](#) ()=default
Default virtual Destructor.
- [detail::RoutingPacket GetCurrentTable](#) () override
Create a Routing Table using the tokens that have been received.

Additional Inherited Members

7.64.1 Detailed Description

A [RoutingMasterPolicy](#) which evenly distributes Sequence IDs to all receivers. If an uneven number of tokens have been received, extra tokens are stored for the next table update.

Definition at line 11 of file RoundRobin_policy.cc.

7.64.2 Constructor & Destructor Documentation

7.64.2.1 artdaq::RoundRobinPolicy::RoundRobinPolicy (`fhicl::ParameterSet ps`) [inline], [explicit]

[RoundRobinPolicy](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure RoundRobinPolicy
-----------------	---

[RoundRobinPolicy](#) accepts no Parameters at this time.

Definition at line 20 of file RoundRobin_policy.cc.

7.64.3 Member Function Documentation

7.64.3.1 detail::RoutingPacket artdaq::RoundRobinPolicy::GetCurrentTable () [override], [virtual]

Create a Routing Table using the tokens that have been received.

Returns

A [detail::RoutingPacket](#) containing the table update

[RoundRobinPolicy](#) will go through the list of receivers as many times as it can, until one or more receivers have no tokens. It always does full "turns" through the receiver list.

Implements [artdaq::RoutingMasterPolicy](#).

Definition at line 38 of file RoundRobin_policy.cc.

The documentation for this class was generated from the following file:

- artdaq/artdaq/Application/Routing/RoundRobin_policy.cc

7.65 artdaq::detail::RoutingAckPacket Struct Reference

A [RoutingAckPacket](#) contains the rank of the table receiver, plus the first and last sequence IDs in the Routing Table (for verification)

```
#include <artdaq/DAQrate/detail/RoutingPacket.hh>
```

Public Attributes

- int `rank`

The rank from which the [RoutingAckPacket](#) came.
- Fragment::sequence_id_t `first_sequence_id`

The first sequence ID in the received RoutingPacket.

- Fragment::sequence_id_t **last_sequence_id**

The last sequence ID in the received RoutingPacket.

7.65.1 Detailed Description

A [RoutingAckPacket](#) contains the rank of the table receiver, plus the first and last sequence IDs in the Routing Table (for verification)

Definition at line 81 of file RoutingPacket.hh.

The documentation for this struct was generated from the following file:

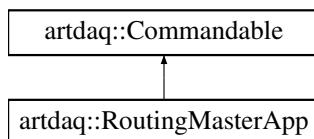
- artdaq/artdaq/DAQrate/detail/RoutingPacket.hh

7.66 artdaq::RoutingMasterApp Class Reference

[RoutingMasterApp](#) is an [artdaq::Commandable](#) derived class which controls the [RoutingMasterCore](#) state machine.

```
#include <artdaq/Application/RoutingMasterApp.hh>
```

Inheritance diagram for artdaq::RoutingMasterApp:



Public Member Functions

- [RoutingMasterApp](#) (int rank, std::string name)
RoutingMasterApp Constructor.
- [RoutingMasterApp](#) ([RoutingMasterApp](#) const &)=delete
Copy Constructor is deleted.
- virtual ~[RoutingMasterApp](#) ()=default
Default Destructor.
- [RoutingMasterApp](#) & [operator=](#) ([RoutingMasterApp](#) const &)=delete
Copy Assignment Operator is deleted.
- bool [do_initialize](#) (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp) override
Initialize the RoutingMasterCore.
- bool [do_start](#) (art::RunID id, uint64_t timeout, uint64_t timestamp) override
Start the RoutingMasterCore.
- bool [do_stop](#) (uint64_t timeout, uint64_t timestamp) override
Stop the RoutingMasterCore.
- bool [do_pause](#) (uint64_t timeout, uint64_t timestamp) override
Pause the RoutingMasterCore.
- bool [do_resume](#) (uint64_t timeout, uint64_t timestamp) override
Resume the RoutingMasterCore.

- bool `do_shutdown` (uint64_t timeout) override
Shutdown the RoutingMasterCore.
- bool `do_soft_initialize` (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp) override
Soft-Initialize the RoutingMasterCore.
- bool `do_reinitialize` (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp) override
Reinitialize the RoutingMasterCore.
- void `BootedEnter` () override
Action taken upon entering the "Booted" state.
- std::string `report` (std::string const &) const override
If which is "transition_status", report the status of the last transition. Otherwise pass through to AggregatorCore.

Additional Inherited Members

7.66.1 Detailed Description

`RoutingMasterApp` is an `artdaq::Commandable` derived class which controls the `RoutingMasterCore` state machine.

Definition at line 17 of file `RoutingMasterApp.hh`.

7.66.2 Constructor & Destructor Documentation

7.66.2.1 `artdaq::RoutingMasterApp::RoutingMasterApp (int rank, std::string name)`

`RoutingMasterApp` Constructor.

Parameters

<code>rank</code>	Rank of this RoutingMaster
<code>name</code>	Friendly name of this application instance (MessageFacility Category)

Default constructor.

Definition at line 6 of file `RoutingMasterApp.cc`.

7.66.3 Member Function Documentation

7.66.3.1 `void artdaq::RoutingMasterApp::BootedEnter () [override], [virtual]`

Action taken upon entering the "Booted" state.

This resets the `RoutingMasterCore` pointer

Reimplemented from `artdaq::Commandable`.

Definition at line 156 of file `RoutingMasterApp.cc`.

7.66.3.2 `bool artdaq::RoutingMasterApp::do_initialize (fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp) [override], [virtual]`

Initialize the `RoutingMasterCore`.

Parameters

<i>pset</i>	ParameterSet used to configure the RoutingMasterCore
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [ardaq::Commandable](#).

Definition at line 13 of file RoutingMasterApp.cc.

7.66.3.3 bool artdaq::RoutingMasterApp::do_pause (uint64_t *timeout*, uint64_t *timestamp*) [override], [virtual]

Pause the [RoutingMasterCore](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [ardaq::Commandable](#).

Definition at line 81 of file RoutingMasterApp.cc.

7.66.3.4 bool artdaq::RoutingMasterApp::do_reinitialize (fhicl::ParameterSet const & *pset*, uint64_t *timeout*, uint64_t *timestamp*) [override], [virtual]

Reinitialize the [RoutingMasterCore](#).

Parameters

<i>pset</i>	ParameterSet used to configure the RoutingMasterCore
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [ardaq::Commandable](#).

Definition at line 144 of file RoutingMasterApp.cc.

7.66.3.5 bool artdaq::RoutingMasterApp::do_resume (uint64_t *timeout*, uint64_t *timestamp*) [override], [virtual]

Resume the [RoutingMasterCore](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 102 of file [RoutingMasterApp.cc](#).

7.66.3.6 `bool artdaq::RoutingMasterApp::do_shutdown (uint64_t timeout) [override], [virtual]`

Shutdown the [RoutingMasterCore](#).

Parameters

<i>timeout</i>	Timeout for transition
----------------	--

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 119 of file [RoutingMasterApp.cc](#).

7.66.3.7 `bool artdaq::RoutingMasterApp::do_soft_initialize (fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp) [override], [virtual]`

Soft-Initialize the [RoutingMasterCore](#).

Parameters

<i>pset</i>	ParameterSet used to configure the RoutingMasterCore
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 131 of file [RoutingMasterApp.cc](#).

7.66.3.8 `bool artdaq::RoutingMasterApp::do_start (art::RunID id, uint64_t timeout, uint64_t timestamp) [override], [virtual]`

Start the [RoutingMasterCore](#).

Parameters

<i>id</i>	Run ID of new run
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 35 of file RoutingMasterApp.cc.

7.66.3.9 bool artdaq::RoutingMasterApp::do_stop (uint64_t *timeout*, uint64_t *timestamp*) [override], [virtual]

Stop the [RoutingMasterCore](#).

Parameters

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 59 of file RoutingMasterApp.cc.

7.66.3.10 RoutingMasterApp& artdaq::RoutingMasterApp::operator= (RoutingMasterApp const &) [delete]

Copy Assignment Operator is deleted.

Returns

[RoutingMasterApp](#) copy

7.66.3.11 std::string artdaq::RoutingMasterApp::report (std::string const & *which*) const [override], [virtual]

If *which* is "transition_status", report the status of the last transition. Otherwise pass through to AggregatorCore.

Parameters

<i>which</i>	What to report on
--------------	-------------------

Returns

Report string. Empty for unknown "which" parameter

Reimplemented from [artdaq::Commandable](#).

Definition at line 167 of file RoutingMasterApp.cc.

The documentation for this class was generated from the following files:

- artdaq/artdaq/Application/RoutingMasterApp.hh
- artdaq/artdaq/Application/RoutingMasterApp.cc

7.67 artdaq::RoutingMasterCore Class Reference

`RoutingMasterCore` implements the state machine for the `RoutingMaster` artdaq application. `RoutingMasterCore` collects tokens from receivers, and at regular intervals uses these tokens to build Routing Tables that are sent to the senders.

```
#include <artdaq/Application/RoutingMasterCore.hh>
```

Public Member Functions

- `RoutingMasterCore` (int rank, std::string name)
RoutingMasterCore Constructor.
- `RoutingMasterCore` (`RoutingMasterCore` const &)=delete
Copy Constructor is deleted.
- `~RoutingMasterCore` ()
- `RoutingMasterCore & operator=` (`RoutingMasterCore` const &)=delete
Copy Assignment operator is deleted.
- `bool initialize` (fhicl::ParameterSet const &pset, uint64_t, uint64_t)
Processes the initialize request.
- `bool start` (art::RunID id, uint64_t, uint64_t)
Starts the RoutingMasterCore.
- `bool stop` (uint64_t, uint64_t)
Stops the RoutingMasterCore.
- `bool pause` (uint64_t, uint64_t)
Pauses the RoutingMasterCore.
- `bool resume` (uint64_t, uint64_t)
Resumes the RoutingMasterCore.
- `bool shutdown` (uint64_t)
Shuts Down the RoutingMasterCore.
- `bool soft_initialize` (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp)
Soft-Initializes the RoutingMasterCore.
- `bool reinitialize` (fhicl::ParameterSet const &pset, uint64_t timeout, uint64_t timestamp)
Reinitializes the RoutingMasterCore.
- `size_t process_event_table` ()
Main loop of the RoutingMasterCore. Determines when to send the next table update, asks the RoutingMasterPolicy for the table to send, and sends it.
- `void send_event_table` (detail::RoutingPacket table)
Sends a detail::RoutingPacket to the table receivers.
- `std::string report` (std::string const &) const
Send a report on the current status of the RoutingMasterCore.

Static Public Attributes

- static const std::string [TABLE_UPDATES_STAT_KEY](#)
Key for Table Update count MonitoredQuantity.
- static const std::string [TOKENS RECEIVED_STAT_KEY](#)
Key for the Tokens Received MonitoredQuantity.

7.67.1 Detailed Description

[RoutingMasterCore](#) implements the state machine for the RoutingMaster artdaq application. [RoutingMasterCore](#) collects tokens from receivers, and at regular intervals uses these tokens to build Routing Tables that are sent to the senders.

Definition at line 31 of file [RoutingMasterCore.hh](#).

7.67.2 Constructor & Destructor Documentation

7.67.2.1 artdaq::RoutingMasterCore::RoutingMasterCore (int rank, std::string name)

[RoutingMasterCore](#) Constructor.

Parameters

<i>rank</i>	Rank of the RoutingMaster
<i>name</i>	Friendly name for the RoutingMaster

Definition at line 29 of file [RoutingMasterCore.cc](#).

7.67.2.2 artdaq::RoutingMasterCore::~RoutingMasterCore ()

Destructor.

Definition at line 46 of file [RoutingMasterCore.cc](#).

7.67.3 Member Function Documentation

7.67.3.1 bool artdaq::RoutingMasterCore::initialize (fhiCL::ParameterSet const & pset, uint64_t , uint64_t)

Processes the initialize request.

Parameters

<i>pset</i>	ParameterSet used to configure the RoutingMasterCore
-------------	--

Returns

Whether the initialize attempt succeeded

- * RoutingMasterCore accepts the following Parameters:
- * "daq" (REQUIRED): FHiCL table containing DAQ configuration
- * "policy" (REQUIRED): FHiCL table containing the RoutingMasterPolicy configuration
- * "policy" (Default: ""): Name of the RoutingMasterPolicy plugin to load
- * "rt_priority" (Default: 0): Unix process priority to assign to RoutingMasterCore
- * "sender_ranks" (REQUIRED): List of ranks (integers) for the senders (that receive table updates)

```

*   "table_update_interval_ms" (Default: 1000): Maximum amount of time between table updates
*   "senders_send_by_send_count" (Default: false): If true, senders will use the current send count to lookup rou
*   "table_ack_retry_count" (Default: 5): The number of times the table will be resent while waiting for acknowled
*   "routing_token_port" (Default: 35555): The port on which to listen for RoutingToken packets
*   "table_update_port" (Default: 35556): The port on which to send table updates
*   "table_acknowledge_port" (Default: 35557): The port on which to listen for RoutingAckPacket datagrams
*   "table_update_address" (Default: "227.128.12.28"): Multicast address to send table updates to
*   "routing_master_hostname" (Default: "localhost"): Hostname to send table updates from
*   "metrics": FHiCL table containing configuration for MetricManager
*

```

Definition at line 52 of file `RoutingMasterCore.cc`.

7.67.3.2 `RoutingMasterCore& artdaq::RoutingMasterCore::operator= (RoutingMasterCore const &)` [delete]

Copy Assignment operator is deleted.

Returns

`RoutingMasterCore` copy

7.67.3.3 `bool artdaq::RoutingMasterCore::pause (uint64_t , uint64_t)`

Pauses the `RoutingMasterCore`.

Returns

True if no exception

Definition at line 183 of file `RoutingMasterCore.cc`.

7.67.3.4 `size_t artdaq::RoutingMasterCore::process_event_table ()`

Main loop of the `RoutingMasterCore`. Determines when to send the next table update, asks the `RoutingMasterPolicy` for the table to send, and sends it.

Returns

Number of table updates sent

Definition at line 225 of file `RoutingMasterCore.cc`.

7.67.3.5 `bool artdaq::RoutingMasterCore::reinitialize (fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp)`

Reinitializes the `RoutingMasterCore`.

Parameters

<code>pset</code>	ParameterSet for configuring <code>RoutingMasterCore</code>
-------------------	---

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Returns initialize status

Definition at line 217 of file RoutingMasterCore.cc.

7.67.3.6 std::string artdaq::RoutingMasterCore::report (std::string const &) const

Send a report on the current status of the [RoutingMasterCore](#).

Returns

A string containing the report on the current status of the [RoutingMasterCore](#)

Definition at line 599 of file RoutingMasterCore.cc.

7.67.3.7 bool artdaq::RoutingMasterCore::resume (uint64_t , uint64_t)

Resumes the [RoutingMasterCore](#).

Returns

True if no exception

Definition at line 192 of file RoutingMasterCore.cc.

7.67.3.8 void artdaq::RoutingMasterCore::send_event_table (detail::RoutingPacket table)

Sends a [detail::RoutingPacket](#) to the table receivers.

Parameters

<i>table</i>	The detail::RoutingPacket to send
--------------	---

`send_event_table` checks the table update socket and the acknowledge socket before sending the table update the first time. It then enters a loop where it sends the table update, then waits for acknowledgement packets. It keeps track of which senders have sent their acknowledgement packets, and discards duplicate acks. It leaves this loop once all senders have sent a valid acknowledgement packet.

Definition at line 307 of file RoutingMasterCore.cc.

7.67.3.9 bool artdaq::RoutingMasterCore::shutdown (uint64_t)

Shuts Down the [RoutingMasterCore](#).

Returns

If the shutdown was successful

Definition at line 201 of file RoutingMasterCore.cc.

7.67.3.10 bool artdaq::RoutingMasterCore::soft_initialize (`fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp`)

Soft-Initializes the [RoutingMasterCore](#).

Parameters

<i>pset</i>	ParameterSet for configuring RoutingMasterCore
<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

Returns

Returns initialize status

Definition at line 209 of file RoutingMasterCore.cc.

7.67.3.11 bool artdaq::RoutingMasterCore::start (art::RunID *id*, uint64_t , uint64_t)

Start the [RoutingMasterCore](#).

Parameters

<i>id</i>	Run ID of the current run
-----------	---------------------------

Returns

True if no exception

Definition at line 157 of file RoutingMasterCore.cc.

7.67.3.12 bool artdaq::RoutingMasterCore::stop (uint64_t , uint64_t)

Stops the [RoutingMasterCore](#).

Returns

True if no exception

Definition at line 174 of file RoutingMasterCore.cc.

The documentation for this class was generated from the following files:

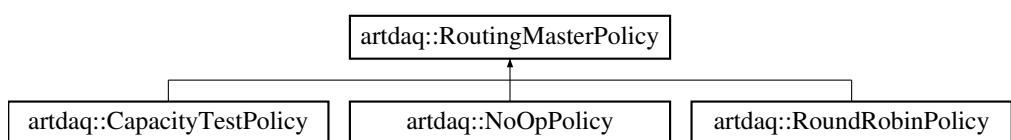
- artdaq/artdaq/Application/RoutingMasterCore.hh
- artdaq/artdaq/Application/RoutingMasterCore.cc

7.68 artdaq::RoutingMasterPolicy Class Reference

The interface through which [RoutingMasterCore](#) obtains Routing Tables using received Routing Tokens.

```
#include <artdaq/Application/Routing/RoutingMasterPolicy.hh>
```

Inheritance diagram for artdaq::RoutingMasterPolicy:



Public Member Functions

- `RoutingMasterPolicy (fhicl::ParameterSet ps)`
`RoutingMasterPolicy` Constructor.
- virtual `~RoutingMasterPolicy ()=default`
`Default virtual Destructor.`
- virtual `detail::RoutingPacket GetCurrentTable ()=0`
`Generate a Routing Table using received tokens.`
- `size_t GetReceiverCount () const`
`Get the number of configured receivers.`
- `size_t GetMaxNumberOfTokens () const`
`Get the largest number of tokens that the RoutingMasterPolicy has seen at any one time.`
- void `AddReceiverToken (int rank, unsigned new_slots_free)`
`Add a token to the token list.`
- void `Reset ()`
`Reset the policy, setting the next sequence ID to be used to 0.`

Protected Member Functions

- `std::unique_ptr< std::deque< int >> getTokensSnapshot ()`
`Gets the current token list, used for building Routing Tables.`
- void `addUnusedTokens (std::unique_ptr< std::deque< int >> tokens)`
`If necessary, return unused tokens to the token list, for subsequent updates.`

Protected Attributes

- `Fragment::sequence_id_t next_sequence_id_`
`The next sequence ID to be assigned.`

7.68.1 Detailed Description

The interface through which `RoutingMasterCore` obtains Routing Tables using received Routing Tokens.

Definition at line 18 of file `RoutingMasterPolicy.hh`.

7.68.2 Constructor & Destructor Documentation

7.68.2.1 artdaq::RoutingMasterPolicy::RoutingMasterPolicy (`fhicl::ParameterSet ps`) [explicit]

`RoutingMasterPolicy` Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure the <code>RoutingMasterPolicy</code>
-----------------	---

* `RoutingMasterPolicy` accepts the following Parameters:

* "receiver_ranks" (REQUIRED): A list of integers indicating the ranks that the `RoutingMasterPolicy` should expect
*

Definition at line 4 of file `RoutingMasterPolicy.cc`.

7.68.3 Member Function Documentation

7.68.3.1 `void artdaq::RoutingMasterPolicy::AddReceiverToken(int rank, unsigned new_slots_free)`

Add a token to the token list.

Parameters

<i>rank</i>	Rank that the token is from
<i>new_slots_free</i>	Number of slots that are now free (should usually be 1)

Definition at line 13 of file RoutingMasterPolicy.cc.

7.68.3.2 `virtual detail::RoutingPacket artdaq::RoutingMasterPolicy::GetCurrentTable() [pure virtual]`

Generate a Routing Table using received tokens.

Returns

A [detail::RoutingPacket](#) containing the Routing Table

This function is pure virtual, it should be overridden by derived classes.

Implemented in [artdaq::CapacityTestPolicy](#), [artdaq::RoundRobinPolicy](#), and [artdaq::NoOpPolicy](#).

7.68.3.3 `size_t artdaq::RoutingMasterPolicy::GetMaxNumberOfTokens() const [inline]`

Get the largest number of tokens that the [RoutingMasterPolicy](#) has seen at any one time.

Returns

The largest number of tokens that the [RoutingMasterPolicy](#) has seen at any one time

Definition at line 54 of file RoutingMasterPolicy.hh.

7.68.3.4 `size_t artdaq::RoutingMasterPolicy::GetReceiverCount() const [inline]`

Get the number of configured receivers.

Returns

The size of the receiver_ranks list

Definition at line 48 of file RoutingMasterPolicy.hh.

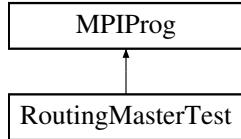
The documentation for this class was generated from the following files:

- [artdaq/artdaq/Application/Routing/RoutingMasterPolicy.hh](#)
- [artdaq/artdaq/Application/Routing/RoutingMasterPolicy.cc](#)

7.69 RoutingMasterTest Class Reference

The [RoutingMasterTest](#) class runs the routing_master test.

Inheritance diagram for RoutingMasterTest:



Public Member Functions

- [RoutingMasterTest \(int argc, char *argv\[\]\)](#)
RoutingMasterTest Constructor.
- [void go \(\)](#)
Start the test, using the role assigned.
- [void generate_tokens \(\)](#)
Generate tokens and send them to the Routing Master.
- [void routing_master \(\)](#)
Load a RoutingMasterCore instance, receive tokens from the token generators, and send table updates to the table receivers.
- [void table_receiver \(\)](#)
Receive Routing Tables from the Routing Master and send acknowledgement packets back.
- [fhicl::ParameterSet getPset \(int argc, char *argv\[\]\) const](#)
Parse the command line arguments and load a configuration FHiCL file.

Additional Inherited Members

7.69.1 Detailed Description

The [RoutingMasterTest](#) class runs the routing_master test.

Definition at line 79 of file [routing_master.cc](#).

7.69.2 Constructor & Destructor Documentation

7.69.2.1 [RoutingMasterTest::RoutingMasterTest \(int argc, char * argv\[\] \)](#)

[RoutingMasterTest](#) Constructor.

Parameters

<i>argc</i>	Number of arguments
<i>argv</i>	Array of argument strings

The configuration file should contains the following Parameters: "daq" (REQUIRED): DAQ configuration "policy" (REQUIRED): The RoutingMasterPolicy configuration "receiver_ranks" (REQUIRED): Ranks of the Table Receivers "token_count" (Default: 1000): Number of tokens to generate "token_interval_us" (Default: 5000): Delay between tokens

The configuration file is also passed to RoutingMasterCore, see that class for required configuration parameters

Definition at line 150 of file [routing_master.cc](#).

7.69.3 Member Function Documentation

7.69.3.1 fhicl::ParameterSet RoutingMasterTest::getPset (int argc, char * argv[]) const

Parse the command line arguments and load a configuration FHiCL file.

Parameters

<i>argc</i>	Number of arguments
<i>argv</i>	Array of argument strings

Returns

ParameterSet used to configure the test

Definition at line 180 of file routing_master.cc.

The documentation for this class was generated from the following file:

- artdaq/proto/routing_master.cc

7.70 artdaq::detail::RoutingPacketEntry Struct Reference

A row of the Routing Table.

```
#include <artdaq/DAQrate/detail/RoutingPacket.hh>
```

Public Member Functions

- [RoutingPacketEntry \(\)](#)
Default Constructor.
- [RoutingPacketEntry \(Fragment::sequence_id_t seq, int rank\)](#)
Construct a [RoutingPacketEntry](#) with the given sequence ID and destination rank.

Public Attributes

- Fragment::sequence_id_t [sequence_id](#)
The sequence ID of the [RoutingPacketEntry](#).
- int [destination_rank](#)
The destination rank for this sequence ID.

7.70.1 Detailed Description

A row of the Routing Table.

Definition at line 35 of file RoutingPacket.hh.

7.70.2 Constructor & Destructor Documentation

7.70.2.1 `artdaq::detail::RoutingPacketEntry::RoutingPacketEntry (Fragment::sequence_id_t seq, int rank) [inline]`

Construct a [RoutingPacketEntry](#) with the given sequence ID and destination rank.

Parameters

<i>seq</i>	The sequence ID of the RoutingPacketEntry
<i>rank</i>	The destination rank for this sequence ID

Definition at line 46 of file RoutingPacket.hh.

The documentation for this struct was generated from the following file:

- artdaq/artdaq/DAQrate/detail/RoutingPacket.hh

7.71 artdaq::detail::RoutingPacketHeader Struct Reference

The header of the Routing Table, containing the magic bytes and the number of entries.

```
#include <artdaq/DAQrate/detail/RoutingPacket.hh>
```

Public Member Functions

- [RoutingPacketHeader \(RoutingMasterMode m, size_t n\)](#)
Construct a [RoutingPacketHeader](#) declaring a given number of entries.
- [RoutingPacketHeader \(\)](#)
Default Constructor.

Public Attributes

- [uint32_t header](#)
Magic bytes to make sure the packet wasn't garbled.
- [RoutingMasterMode mode](#)
The current mode of the RoutingMaster.
- [size_t nEntries](#)
The number of [RoutingPacketEntries](#) in the [RoutingPacket](#).

7.71.1 Detailed Description

The header of the Routing Table, containing the magic bytes and the number of entries.

Definition at line 60 of file RoutingPacket.hh.

7.71.2 Constructor & Destructor Documentation

7.71.2.1 artdaq::detail::RoutingPacketHeader::RoutingPacketHeader ([RoutingMasterMode m, size_t n](#)) [inline], [explicit]

Construct a [RoutingPacketHeader](#) declaring a given number of entries.

Parameters

<i>m</i>	The RoutingMasterMode that senders are supposed to be operating in
<i>n</i>	The number of RoutingPacketEntries in the associated RoutingPacket

Definition at line 71 of file RoutingPacket.hh.

The documentation for this struct was generated from the following file:

- artdaq/artdaq/DAQrate/detail/RoutingPacket.hh

7.72 artdaq::detail::RoutingToken Struct Reference

The [RoutingToken](#) contains the magic bytes, the rank of the token sender, and the number of slots free. This is a TCP message, so additional verification is not necessary.

```
#include <artdaq/DAQrate/detail/RoutingPacket.hh>
```

Public Attributes

- `uint32_t header`
The magic bytes that help validate the [RoutingToken](#).
- `int rank`
The rank from which the [RoutingToken](#) came.
- `unsigned new_slots_free`
The number of slots free in the token sender (usually 1)

7.72.1 Detailed Description

The [RoutingToken](#) contains the magic bytes, the rank of the token sender, and the number of slots free. This is a TCP message, so additional verification is not necessary.

Definition at line 98 of file RoutingPacket.hh.

The documentation for this struct was generated from the following file:

- artdaq/artdaq/DAQrate/detail/RoutingPacket.hh

7.73 artdaq::RTIDDS Class Reference

DDS Transport Implementation.

```
#include <artdaq/RTIDDS/RTIDDS.hh>
```

Classes

- class [OctetsListener](#)
A class that reads data from DDS.

Public Types

- enum [IOType](#) { **reader**, **writer** }

Whether this DDS instance is a reader or a writer.

Public Member Functions

- [RTIDDS](#) (std::string name, [IOType](#) iotype, std::string max_size="1000000")
Construct a [RTIDDS](#) transmitter.
- virtual [~RTIDDS](#) ()=default
Default virtual Destructor.
- void [copyFragmentToDDS_](#) (artdaq::Fragment &fragment)
Copy a Fragment to DDS.
- void [moveFragmentToDDS_](#) (artdaq::Fragment &&fragment)
Move a Fragment to DDS.

Public Attributes

- [OctetsListener octets_listener_](#)

The receiver.

7.73.1 Detailed Description

DDS Transport Implementation.

Definition at line 21 of file RTIDDS.hh.

7.73.2 Constructor & Destructor Documentation

7.73.2.1 artdaq::RTIDDS::RTIDDS (std::string *name*, [IOType](#) *iotype*, std::string *max_size* = "1000000")

Construct a [RTIDDS](#) transmitter.

Parameters

<i>name</i>	Name of the module
<i>iotype</i>	Direction of transmission
<i>max_size</i>	Maximum size to transmit

Definition at line 9 of file RTIDDS.cc.

7.73.3 Member Function Documentation

7.73.3.1 void artdaq::RTIDDS::copyFragmentToDDS_ (artdaq::Fragment & *fragment*)

Copy a Fragment to DDS.

Parameters

<i>fragment</i>	Fragment to copy
-----------------	------------------

This function may be non-reliable, and induces a memcpy of the Fragment

Definition at line 112 of file RTIDDS.cc.

7.73.3.2 void artdaq::RTIDDS::moveFragmentToDDS_(artdaq::Fragment && *fragment*)

Move a Fragment to DDS.

Parameters

<i>fragment</i>	Fragment to move
-----------------	------------------

This function should be reliable, and minimize copies. Currently implemented via copyFragmentToDDS_

Definition at line 110 of file RTIDDS.cc.

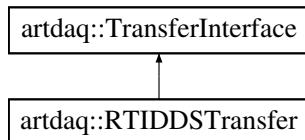
The documentation for this class was generated from the following files:

- artdaq/artdaq/RTIDDS/RTIDDS.hh
- artdaq/artdaq/RTIDDS/RTIDDS.cc

7.74 artdaq::RTIDDSTransfer Class Reference

RTIDDSTransfer is a [TransferInterface](#) implementation plugin that transfers data using RTI DDS.

Inheritance diagram for artdaq::RTIDDSTransfer:



Public Member Functions

- virtual ~RTIDDSTransfer ()=default
RTIDDSTransfer default Destructor.
- RTIDDSTransfer (fhicl::ParameterSet const &ps, [Role role](#))
RTIDDSTransfer Constructor.
- int [receiveFragment](#) (artdaq::Fragment &fragment, size_t receiveTimeout) override
Receive a Fragment using DDS.
- [CopyStatus copyFragment](#) (artdaq::Fragment &fragment, size_t send_timeout_usec=std::numeric_limits< size_t >::max()) override
Copy a Fragment to the destination.
- [CopyStatus moveFragment](#) (artdaq::Fragment &&fragment, size_t send_timeout_usec=std::numeric_limits< size_t >::max()) override
Move a Fragment to the destination.

Additional Inherited Members

7.74.1 Detailed Description

`RTIDDSTransfer` is a [TransferInterface](#) implementation plugin that transfers data using RTI DDS.

Definition at line 22 of file `RTIDDS_transfer.cc`.

7.74.2 Constructor & Destructor Documentation

7.74.2.1 artdaq::RTIDDSTransfer::RTIDDSTransfer (`fhcl::ParameterSet const & ps, Role role`) [inline]

`RTIDDSTransfer` Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure <code>RTIDDSTransfer</code>
<code>role</code>	Role of this <code>RTIDDSTransfer</code> instance (kSend or kReceive)

`RTIDDSTransfer` only requires the Parameters for configuring a [TransferInterface](#)

Definition at line 37 of file `RTIDDS_transfer.cc`.

7.74.3 Member Function Documentation

7.74.3.1 artdaq::TransferInterface::CopyStatus artdaq::RTIDDSTransfer::copyFragment (`artdaq::Fragment & fragment, size_t send_timeout_usec = std::numeric_limits<size_t>::max()`) [override], [virtual]

Copy a Fragment to the destination.

Parameters

<code>fragment</code>	Fragment to copy
<code>send_timeout_usec</code>	Timeout for send, in microseconds. Default <code>size_t::MAX_VALUE</code>

Returns

`CopyStatus` detailing result of copy

Implements [artdaq::TransferInterface](#).

Definition at line 122 of file `RTIDDS_transfer.cc`.

7.74.3.2 artdaq::TransferInterface::CopyStatus artdaq::RTIDDSTransfer::moveFragment (`artdaq::Fragment && fragment, size_t send_timeout_usec = std::numeric_limits<size_t>::max()`) [override], [virtual]

Move a Fragment to the destination.

Parameters

<i>fragment</i>	Fragment to move
<i>send_timeout_usec</i>	Timeout for send, in microseconds. Default size_t::MAX_VALUE

Returns

CopyStatus detailing result of copy

Implements [artdaq::TransferInterface](#).

Definition at line 113 of file RTIDDS_transfer.cc.

7.74.3.3 int artdaq::RTIDDSTransfer::receiveFragment (artdaq::Fragment & *fragment*, size_t *receiveTimeout*) [override], [virtual]

Receive a Fragment using DDS.

Parameters

<i>out</i>	<i>fragment</i>	Received Fragment
	<i>receiveTimeout</i>	Timeout for receive, in microseconds

Returns

Rank of sender or RECV_TIMEOUT

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 75 of file RTIDDS_transfer.cc.

The documentation for this class was generated from the following file:

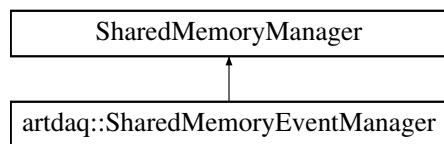
- artdaq/artdaq/TransferPlugins/RTIDDS_transfer.cc

7.75 artdaq::SharedMemoryEventManager Class Reference

The [SharedMemoryEventManager](#) is a SharedMemoryManger which tracks events as they are built.

```
#include <artdaq/DAQrate/SharedMemoryEventManager.hh>
```

Inheritance diagram for artdaq::SharedMemoryEventManager:



Public Types

- `typedef RawEvent::run_id_t run_id_t`

- `typedef RawEvent::run_id_t subrun_id_t`
Copy RawEvent::run_id_t into local scope.
- `typedef RawEvent::subrun_id_t subrun_id_t`
Copy RawEvent::subrun_id_t into local scope.
- `typedef Fragment::sequence_id_t sequence_id_t`
Copy Fragment::sequence_id_t into local scope.
- `typedef std::map<sequence_id_t, RawEvent_ptr> EventMap`
An EventMap is a map of RawEvent_ptr objects, keyed by sequence ID.

Public Member Functions

- `SharedMemoryEventManager (fhicl::ParameterSet pset, fhicl::ParameterSet art_pset)`
SharedMemoryEventManager Constructor.
- `virtual ~SharedMemoryEventManager ()`
SharedMemoryEventManager Destructor.
- `bool AddFragment (FragmentPtr frag, size_t timeout_usec, FragmentPtr &outfrag)`
Copy a Fragment into the SharedMemoryEventManager.
- `RawDataType * WriteFragmentHeader (detail::RawFragmentHeader frag, bool dropIfNoBuffersAvailable=false)`
Get a pointer to a reserved memory area for the given Fragment header.
- `void DoneWritingFragment (detail::RawFragmentHeader frag)`
Used to indicate that the given Fragment is now completely in the buffer. Will check for buffer completeness, and unset the pending flag.
- `size_t GetInactiveEventCount ()`
Returns the number of buffers which have no data.
- `size_t GetIncompleteEventCount ()`
Returns the number of buffers which contain data but are not yet complete.
- `size_t GetPendingEventCount ()`
Returns the number of events which are complete but waiting on lower sequenced events to finish.
- `size_t GetLockedBufferCount ()`
Returns the number of buffers currently owned by this manager.
- `size_t GetArtEventCount ()`
Returns the number of events sent to art this subrun.
- `size_t GetFragmentCount (Fragment::sequence_id_t seqID, Fragment::type_t type=Fragment::InvalidFragmentType)`
Get the count of Fragments of a given type in an event.
- `void RunArt (std::shared_ptr<art_config_file> config_file, pid_t &pid_out)`
Run an art instance, recording the return codes and restarting it until the end flag is raised.
- `void StartArt ()`
Start all the art processes.
- `pid_t StartArtProcess (fhicl::ParameterSet pset)`
Start one art process.
- `void ShutdownArtProcesses (std::set<pid_t> pids)`
Shutdown a set of art processes.
- `void ReconfigureArt (fhicl::ParameterSet art_pset, run_id_t newRun=0, int n_art_processes=-1)`
Restart all art processes, using the given fhicl code to configure the new art processes.
- `bool endOfData ()`
Indicate that the end of input has been reached to the art processes.

- void [startRun \(run_id_t runID\)](#)
Start a Run.
- void [startSubrun \(\)](#)
Start a new Subrun, incrementing the subrun number.
- [run_id_t runID \(\) const](#)
Get the current Run number.
- [subrun_id_t subrunID \(\) const](#)
Get the current subrun number.
- bool [endRun \(\)](#)
Send an EndOfRunFragment to the art thread.
- bool [endSubrun \(\)](#)
Send an EndOfSubRunFragment to the art thread.
- void [sendMetrics \(\)](#)
Send metrics to the MetricManager, if one has been instantiated in the application.
- void [setRequestMode \(detail::RequestMessageMode mode\)](#)
Set the RequestMessageMode for all outgoing data requests.
- void [setOverwrite \(bool overwrite\)](#)
Set the overwrite flag (non-reliable data transfer) for the Shared Memory.
- void [SetInitFragment \(FragmentPtr frag\)](#)
Set the stored Init fragment, if one has not yet been set already.
- uint32_t [GetBroadcastKey \(\)](#)
Gets the shared memory key of the broadcast SharedMemoryManager.
- RawDataType * [GetDroppedDataAddress \(\)](#)
Gets the address of the "dropped data" fragment. Used for testing.

7.75.1 Detailed Description

The [SharedMemoryEventManager](#) is a SharedMemoryManger which tracks events as they are built.

Definition at line 59 of file SharedMemoryEventManager.hh.

7.75.2 Constructor & Destructor Documentation

7.75.2.1 [artdaq::SharedMemoryEventManager::SharedMemoryEventManager \(fhicl::ParameterSet pset, fhicl::ParameterSet art_pset \)](#)

[SharedMemoryEventManager](#) Constructor.

Parameters

pset	ParameterSet used to configure SharedMemoryEventManager
art_pset	ParameterSet used to configure art

* SharedMemoryEventManager accepts the following Parameters:
*
* "buffer_count" REQUIRED: Number of events in the Shared Memory (incomplete + pending art)
* "max_event_size_bytes" REQUIRED: Maximum event size (all Fragments), in bytes
* OR "max_fragment_size_bytes" REQUIRED: Maximum Fragment size, in bytes
* "stale_buffer_touch_count" (Default: 0x10000): Maximum number of times a buffer may be queried before being marked stale.
* Owner resets this counter every time it touches the buffer.
* "art_analyzer_count" (Default: 1): Number of art processes to start

```
* "expected.fragments_per_event" (REQUIRED): Number of Fragments to expect per event
* "update_run_ids_on_new_fragment" (Default: true): Whether the run and subrun ID of an event should be updated when a new fragment is added
* "incomplete_event_report_interval_ms" (Default: -1): Interval at which an incomplete event report should be written
```

Definition at line 7 of file SharedMemoryEventManager.cc.

7.75.3 Member Function Documentation

7.75.3.1 bool artdaq::SharedMemoryEventManager::AddFragment (FragmentPtr *frag*, size_t *timeout_usec*, FragmentPtr & *outfrag*)

Copy a Fragment into the [SharedMemoryEventManager](#).

Parameters

	<i>frag</i>	FragmentPtr object
	<i>timeout_usec</i>	Timeout for adding Fragment to the Shared Memory
out	<i>outfrag</i>	Rejected Fragment if timeout occurs

Returns

Whether the Fragment was successfully added

Definition at line 111 of file SharedMemoryEventManager.cc.

7.75.3.2 void artdaq::SharedMemoryEventManager::DoneWritingFragment (detail::RawFragmentHeader *frag*)

Used to indicate that the given Fragment is now completely in the buffer. Will check for buffer completeness, and unset the pending flag.

Parameters

<i>frag</i>	Fragment that is now completely in the buffer.
-------------	--

Definition at line 165 of file SharedMemoryEventManager.cc.

7.75.3.3 bool artdaq::SharedMemoryEventManager::endOfData ()

Indicate that the end of input has been reached to the art processes.

Returns

True if the end proceeded correctly

Put the end-of-data marker onto the RawEvent queue (if possible), wait for the reader function to exit, and fill in the reader return value. This scenario returns true. If the end-of-data marker can not be pushed onto the RawEvent queue, false is returned.

Definition at line 394 of file SharedMemoryEventManager.cc.

7.75.3.4 bool artdaq::SharedMemoryEventManager::endRun ()

Send an EndOfRunFragment to the art thread.

Returns

True if enqueue successful

Definition at line 501 of file SharedMemoryEventManager.cc.

7.75.3.5 bool artdaq::SharedMemoryEventManager::endSubrun()

Send an EndOfSubRunFragment to the art thread.

Returns

True if enqueue successful

Definition at line 515 of file SharedMemoryEventManager.cc.

7.75.3.6 size_t artdaq::SharedMemoryEventManager::GetArtEventCount() [inline]

Returns the number of events sent to art this subrun.

Returns

The number of events sent to art this subrun

Definition at line 153 of file SharedMemoryEventManager.hh.

7.75.3.7 uint32_t artdaq::SharedMemoryEventManager::GetBroadcastKey() [inline]

Gets the shared memory key of the broadcast SharedMemoryManager.

Returns

The shared memory key of the broadcast SharedMemoryManager

Definition at line 265 of file SharedMemoryEventManager.hh.

7.75.3.8 RawDataType* artdaq::SharedMemoryEventManager::GetDroppedDataAddress() [inline]

Gets the address of the "dropped data" fragment. Used for testing.

Returns

Pointer to the data payload of the "dropped data" fragment

Definition at line 271 of file SharedMemoryEventManager.hh.

7.75.3.9 size_t artdaq::SharedMemoryEventManager::GetFragmentCount(Fragment::sequence_id_t seqID, Fragment::type_t type = Fragment::InvalidFragmentType)

Get the count of Fragments of a given type in an event.

Parameters

<i>seqID</i>	Sequence ID of Fragments
<i>type</i>	Type of fragments to count. Use InvalidFragmentType to count all fragments (default)

Returns

Number of Fragments in event of given type

Definition at line 185 of file SharedMemoryEventManager.cc.

7.75.3.10 size_t artdaq::SharedMemoryEventManager::GetInactiveEventCount() [inline]

Returns the number of buffers which have no data.

Returns

The number of buffers which have no data

Definition at line 129 of file SharedMemoryEventManager.hh.

7.75.3.11 size_t artdaq::SharedMemoryEventManager::GetIncompleteEventCount() [inline]

Returns the number of buffers which contain data but are not yet complete.

Returns

The number of buffers which contain data but are not yet complete

Definition at line 135 of file SharedMemoryEventManager.hh.

7.75.3.12 size_t artdaq::SharedMemoryEventManager::GetLockedBufferCount() [inline]

Returns the number of buffers currently owned by this manager.

Returns

The number of buffers currently owned by this manager

Definition at line 147 of file SharedMemoryEventManager.hh.

7.75.3.13 size_t artdaq::SharedMemoryEventManager::GetPendingEventCount() [inline]

Returns the number of events which are complete but waiting on lower sequenced events to finish.

Returns

The number of events which are complete but waiting on lower sequenced events to finish

Definition at line 141 of file SharedMemoryEventManager.hh.

7.75.3.14 void artdaq::SharedMemoryEventManager::ReconfigureArt(fhicl::ParameterSet art_pset, run_id_t newRun = 0, int n_art_processes = -1)

Restart all art processes, using the given fhicl code to configure the new art processes.

Parameters

<i>art_pset</i>	ParameterSet used to configure art
<i>newRun</i>	New Run number for reconfigured art
<i>n_art_processes</i>	Number of art processes to start, -1 (default) leaves the number unchanged

Definition at line 370 of file SharedMemoryEventManager.cc.

7.75.3.15 run_id_t artdaq::SharedMemoryEventManager::runID() const [inline]

Get the current Run number.

Returns

The current Run number

Definition at line 219 of file SharedMemoryEventManager.hh.

7.75.3.16 void artdaq::SharedMemoryEventManager::setOverwrite(bool overwrite) [inline]

Set the overwrite flag (non-reliable data transfer) for the Shared Memory.

Parameters

<i>overwrite</i>	Whether to allow the writer to overwrite data that has not yet been read
------------------	--

Definition at line 254 of file SharedMemoryEventManager.hh.

7.75.3.17 void artdaq::SharedMemoryEventManager::setRequestMethod(detail::RequestMessageMode mode) [inline]

Set the RequestMessageMode for all outgoing data requests.

Parameters

<i>mode</i>	Mode to set
-------------	-------------

Definition at line 248 of file SharedMemoryEventManager.hh.

7.75.3.18 void artdaq::SharedMemoryEventManager::ShutdownArtProcesses(std::set< pid_t > pids)

Shutdown a set of art processes.

Parameters

<i>pids</i>	PIDs of the art processes
-------------	---------------------------

Definition at line 289 of file SharedMemoryEventManager.cc.

7.75.3.19 pid_t artdaq::SharedMemoryEventManager::StartArtProcess(fhicl::ParameterSet pset)

Start one art process.

Parameters

<i>pset</i>	ParameterSet to send to this art process
-------------	--

Returns

pid_t of the started process

Definition at line 250 of file SharedMemoryEventManager.cc.

7.75.3.20 void artdaq::SharedMemoryEventManager::startRun (run_id_t runID)

Start a Run.

Parameters

<i>runID</i>	Run number of the new run
--------------	---------------------------

Definition at line 471 of file SharedMemoryEventManager.cc.

7.75.3.21 subrun_id_t artdaq::SharedMemoryEventManager::subrunID () const [inline]

Get the current subrun number.

Returns

The current subrun number

Definition at line 225 of file SharedMemoryEventManager.hh.

7.75.3.22 artdaq::RawDataType * artdaq::SharedMemoryEventManager::WriteFragmentHeader (detail::RawFragmentHeader frag, bool dropIfNoBuffersAvailable = false)

Get a pointer to a reserved memory area for the given Fragment header.

Parameters

<i>frag</i>	Fragment header (contains sequence ID and size information)
<i>dropIfNoBuffersAvailable</i>	Whether to drop the fragment (instead of returning nullptr) when no buffers are available (Default: false)

Returns

Pointer to memory location for Fragment body (Header is copied into buffer here)

Definition at line 131 of file SharedMemoryEventManager.cc.

The documentation for this class was generated from the following files:

- artdaq/artdaq/DAQrate/SharedMemoryEventManager.hh
- artdaq/artdaq/DAQrate/SharedMemoryEventManager.cc

7.76 artdaq::detail::SharedMemoryReader< getDefaultTypes > Struct Template Reference

The [SharedMemoryReader](#) is a class which implements the methods needed by [art::Source](#).

```
#include <artdaq/ArtModules/detail/SharedMemoryReader.hh>
```

Public Member Functions

- [SharedMemoryReader \(SharedMemoryReader const &\)=delete](#)
Copy Constructor is deleted.
- [SharedMemoryReader & operator= \(SharedMemoryReader const &\)=delete](#)
Copy Assignment operator is deleted.
- [SharedMemoryReader \(fhicl::ParameterSet const &ps, art::ProductRegistryHelper &help, art::SourceHelper const &pm\)](#)
SharedMemoryReader Constructor.
- [SharedMemoryReader \(fhicl::ParameterSet const &ps, art::ProductRegistryHelper &help, art::SourceHelper const &pm, art::MasterProductRegistry &\)](#)
SharedMemoryReader Constructor.
- virtual [~SharedMemoryReader \(\)=default](#)
SharedMemoryReader destructor.
- void [closeCurrentFile \(\)](#)
Emulate closing a file. No-Op.
- void [readFile \(std::string const &, art::FileBlock *&fb\)](#)
Emulate opening a file.
- bool [hasMoreData \(\) const](#)
Whether more data is expected from the SharedMemoryReader.
- bool [readNext \(art::RunPrincipal *const &inR, art::SubRunPrincipal *const &inSR, art::RunPrincipal *&outR, art::SubRunPrincipal *&outSR, art::EventPrincipal *&outE\)](#)
Dequeue a RawEvent and declare its Fragment contents to art, creating Run, SubRun, and EventPrincipal objects as necessary.

Public Attributes

- art::SourceHelper const [pmaker](#)
An art::SourceHelper instance.
- std::unique_ptr
< SharedMemoryEventReceiver > [incoming_events](#)
The events from the EventStore.
- double [waiting_time](#)
The amount of time to wait for an event from the queue.
- bool [resume_after_timeout](#)
Whether to resume if the dequeue action times out.
- std::string [pretend_module_name](#)
The module name to store data under.
- std::string [unidentified_instance_name](#)
The name to use for unknown Fragment types.
- bool [shutdownMsgReceived](#)
Whether a shutdown message has been received.

- bool `outputFileCloseNeeded`
If an explicit output file close message is needed.
- size_t `bytesRead`
running total of number of bytes received
- std::unique_ptr
`< SharedMemoryManager > data_shm`
SharedMemoryManager containing data.
- std::unique_ptr
`< SharedMemoryManager > broadcast_shm`
SharedMemoryManager containing broadcasts (control Fragments)
- std::map< Fragment::type_t,
 std::string > `fragment_type_map_`
The Fragment type names that this `SharedMemoryReader` knows about.
- unsigned `readNext_calls_`
The number of times `readNext` has been called.

7.76.1 Detailed Description

```
template<std::map< artdaq::Fragment::type_t, std::string > getDefaultTypes = artdaq::Fragment::MakeSystemTypeMap>struct
artdaq::detail::SharedMemoryReader< getDefaultTypes >
```

The `SharedMemoryReader` is a class which implements the methods needed by `art::Source`.

Definition at line 36 of file `SharedMemoryReader.hh`.

7.76.2 Constructor & Destructor Documentation

```
7.76.2.1 template<std::map< artdaq::Fragment::type_t, std::string > getDefaultTypes = artdaq::Fragment::MakeSystemTypeMap>
artdaq::detail::SharedMemoryReader< getDefaultTypes >::SharedMemoryReader( fhicl::ParameterSet const
& ps, art::ProductRegistryHelper & help, art::SourceHelper const & pm ) [inline]
```

`SharedMemoryReader` Constructor.

Parameters

<code>ps</code>	ParameterSet used for configuring <code>SharedMemoryReader</code>
<code>help</code>	art::ProductRegistryHelper which is used to inform art about different Fragment types
<code>pm</code>	art::SourceHelper used to initialize the SourceHelper member

* `SharedMemoryReader` accepts the following Parameters:
* "waiting_time" (Default: 86400.0): The maximum amount of time to wait for an event from the queue
* "resume_after_timeout" (Default: true): Whether to continue receiving data after a timeout
* "raw_data_label" (Default: "daq"): The label to use for all raw data
* "shared_memory_key" (Default: 0xBEE7): The key for the shared memory segment
*

Definition at line 75 of file `SharedMemoryReader.hh`.

```
7.76.2.2 template<std::map< artdaq::Fragment::type_t, std::string > getDefaultTypes = artdaq::Fragment::MakeSystemTypeMap>
artdaq::detail::SharedMemoryReader< getDefaultTypes >::SharedMemoryReader( fhicl::ParameterSet const
& ps, art::ProductRegistryHelper & help, art::SourceHelper const & pm, art::MasterProductRegistry & ) [inline]
```

`SharedMemoryReader` Constructor.

Parameters

<i>ps</i>	ParameterSet used for configuring SharedMemoryReader
<i>help</i>	art::ProductRegistryHelper which is used to inform art about different Fragment types
<i>pm</i>	art::SourceHelper used to initialize the SourceHelper member

This constructor calls the three-parameter constructor, the art::MasterProductRegistry parameter is discarded.

Definition at line 125 of file SharedMemoryReader.hh.

7.76.3 Member Function Documentation

7.76.3.1 template<std::map< artdaq::Fragment::type_t, std::string > getDefaultTypes = artdaq::Fragment::MakeSystemTypeMap>
bool artdaq::detail::SharedMemoryReader< getDefaultTypes >::hasMoreData() const [inline]

Whether more data is expected from the [SharedMemoryReader](#).

Returns

True unless a shutdown message has been received in readNext

Definition at line 155 of file SharedMemoryReader.hh.

7.76.3.2 template<std::map< artdaq::Fragment::type_t, std::string > getDefaultTypes = artdaq::Fragment::MakeSystemTypeMap>
SharedMemoryReader& artdaq::detail::SharedMemoryReader< getDefaultTypes >::operator= (SharedMemoryReader< getDefaultTypes > const &) [delete]

Copy Assignment operator is deleted.

Returns

[SharedMemoryReader](#) copy

7.76.3.3 template<std::map< artdaq::Fragment::type_t, std::string > getDefaultTypes = artdaq::Fragment::MakeSystemTypeMap>
void artdaq::detail::SharedMemoryReader< getDefaultTypes >::readFile(std::string const &, art::FileBlock *& fb) [inline]

Emulate opening a file.

Parameters

<i>out</i>	<i>fb</i>	art::FileBlock object
------------	-----------	-----------------------

Definition at line 145 of file SharedMemoryReader.hh.

7.76.3.4 template<std::map< artdaq::Fragment::type_t, std::string > getDefaultTypes = artdaq::Fragment::MakeSystemTypeMap>
bool artdaq::detail::SharedMemoryReader< getDefaultTypes >::readNext(art::RunPrincipal *const & *inR*, art::SubRunPrincipal *const & *inSR*, art::RunPrincipal *& *outR*, art::SubRunPrincipal *& *outSR*, art::EventPrincipal *& *outE*) [inline]

Dequeue a RawEvent and declare its Fragment contents to art, creating Run, SubRun, and EventPrincipal objects as necessary.

Parameters

<i>in</i>	<i>inR</i>	Input art::RunPrincipal
<i>in</i>	<i>inSR</i>	Input art::SubRunPrincipal
<i>out</i>	<i>outR</i>	Output art::RunPrincipal
<i>out</i>	<i>outSR</i>	Output art::SubRunPrincipal
<i>out</i>	<i>outE</i>	Output art::EventPrincipal

Returns

Whether an event was returned

Definition at line 167 of file SharedMemoryReader.hh.

The documentation for this struct was generated from the following file:

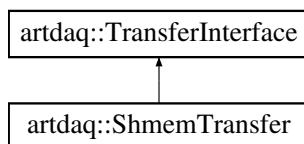
- artdaq/artdaq/ArtModules/detail/SharedMemoryReader.hh

7.77 artdaq::ShmemTransfer Class Reference

A [TransferInterface](#) implementation plugin that transfers data using Shared Memory.

```
#include <artdaq/TransferPlugins/ShmemTransfer.hh>
```

Inheritance diagram for artdaq::ShmemTransfer:



Public Member Functions

- **ShmemTransfer** (fhicl::ParameterSet const &pset, **Role role**)
ShmemTransfer Constructor.
- virtual ~**ShmemTransfer** () noexcept
ShmemTransfer Destructor.
- int **receiveFragment** (Fragment &fragment, size_t receiveTimeout) override
Receive a Fragment from Shared Memory.
- int **receiveFragmentHeader** (detail::RawFragmentHeader &header, size_t receiveTimeout) override
Receive a Fragment Header from the transport mechanism.
- int **receiveFragmentData** (RawDataType *destination, size_t wordCount) override
Receive the body of a Fragment to the given destination pointer.
- **CopyStatus copyFragment** (Fragment &fragment, size_t send_timeout_usec=std::numeric_limits< size_t >-::max()) override
Copy a Fragment to the destination. May be unreliable.
- **CopyStatus moveFragment** (Fragment &&fragment, size_t send_timeout_usec=std::numeric_limits< size_t >-::max()) override
Move a Fragment to the destination.

Additional Inherited Members

7.77.1 Detailed Description

A [TransferInterface](#) implementation plugin that transfers data using Shared Memory.

Definition at line 14 of file ShmemTransfer.hh.

7.77.2 Constructor & Destructor Documentation

7.77.2.1 artdaq::ShmemTransfer::ShmemTransfer (*fhicl::ParameterSet const & pset, Role role*)

[ShmemTransfer](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure ShmemTransfer
<i>role</i>	Role of this ShmemTransfer instance (kSend or kReceive)

- * ShmemTransfer accepts the following Parameters:
- * "shm_key" (Default: hash of uniqueLabel): Key to use for shared memory segment
- *

[ShmemTransfer](#) also requires all Parameters for configuring a [TransferInterface](#)

Definition at line 4 of file Shmem_transfer.cc.

7.77.3 Member Function Documentation

7.77.3.1 artdaq::TransferInterface::CopyStatus artdaq::ShmemTransfer::copyFragment (*Fragment & fragment, size_t send_timeout_usec = std::numeric_limits<size_t>::max()*) [override]

Copy a Fragment to the destination. May be unreliable.

Parameters

<i>fragment</i>	Fragment to copy
<i>send_timeout_usec</i>	Timeout for send, in microseconds (default size_t::MAX_VALUE)

Returns

CopyStatus detailing result of copy

Definition at line 143 of file Shmem_transfer.cc.

7.77.3.2 artdaq::TransferInterface::CopyStatus artdaq::ShmemTransfer::moveFragment (*Fragment && fragment, size_t send_timeout_usec = std::numeric_limits<size_t>::max()*) [override]

Move a Fragment to the destination.

Parameters

<i>fragment</i>	Fragment to move
<i>send_timeout_usec</i>	Timeout for send, in microseconds (default size_t::MAX_VALUE)

Returns

CopyStatus detailing result of move

Definition at line 149 of file Shmem_transfer.cc.

7.77.3.3 int artdaq::ShmemTransfer::receiveFragment (Fragment & *fragment*, size_t *receiveTimeout*) [override]

Receive a Fragment from Shared Memory.

Parameters

<i>out</i>	<i>fragment</i>	Received Fragment
	<i>receiveTimeout</i>	Timeout for receive, in microseconds

Returns

Rank of sender or RECV_TIMEOUT

Definition at line 50 of file Shmem_transfer.cc.

7.77.3.4 int artdaq::ShmemTransfer::receiveFragmentData (RawDataType * *destination*, size_t *wordCount*) [override], [virtual]

Receive the body of a Fragment to the given destination pointer.

Parameters

<i>destination</i>	Pointer to memory region where Fragment data should be stored
<i>wordCount</i>	Number of words of Fragment data to receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implements [ardaq::TransferInterface](#).

Definition at line 129 of file Shmem_transfer.cc.

7.77.3.5 int artdaq::ShmemTransfer::receiveFragmentHeader (detail::RawFragmentHeader & *header*, size_t *receiveTimeout*) [override], [virtual]

Receive a Fragment Header from the transport mechanism.

Parameters

<code>out</code>	<code>header</code>	Received Fragment Header
	<code>receiveTimeout</code>	Timeout for receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implements [artdaq::TransferInterface](#).

Definition at line 90 of file Shmem_transfer.cc.

The documentation for this class was generated from the following files:

- artdaq/artdaq/TransferPlugins/ShmemTransfer.hh
- artdaq/artdaq/TransferPlugins/Shmem_transfer.cc

7.78 ShmRTestFixture Struct Reference

SharedMemoryReader Test Fixture.

Public Member Functions

- [ShmRTestFixture \(\)](#)
ShmRTestFixture Constructor.
- [MPRGlobalTestFixture & gf \(\)](#)
Get a MPRGlobalTestFixture, creating a static instance if necessary.
- [art::ProductRegistryHelper & helper \(\)](#)
Get an art::ProductRegistryHelper, creating a static instance if necessary.
- [art::SourceHelper & source_helper \(\)](#)
Get an art::SourceHelper object, creating a static instance if necessary.
- [uint32_t getKey \(\)](#)
Gets the key for the shared memory segment.
- [uint32_t getBroadcastKey \(\)](#)
Gets the key for the broadcast shared memory segment.
- [artdaq::detail::SharedMemoryReader & reader \(\)](#)
Get an artdaq::detail::SharedMemoryReader object, creating a static instance if necessary.
- [artdaq::SharedMemoryEventManager & writer \(\)](#)
Get the instance of the SharedMemoryEventManager.

7.78.1 Detailed Description

SharedMemoryReader Test Fixture.

Definition at line 178 of file shared_memory_reader_t.cc.

7.78.2 Member Function Documentation

7.78.2.1 `uint32_t ShmRTestFixture::getBroadcastKey() [inline]`

Gets the key for the broadcast shared memory segment.

Returns

Key of the broadcast shared memory segment

Definition at line 262 of file `shared_memory_reader_t.cc`.

7.78.2.2 `uint32_t ShmRTestFixture::getKey() [inline]`

Gets the key for the shared memory segment.

Returns

Key of the shared memory segment

Definition at line 253 of file `shared_memory_reader_t.cc`.

7.78.2.3 `MPRGlobalTestFixture& ShmRTestFixture::gf() [inline]`

Get a `MPRGlobalTestFixture`, creating a static instance if necessary.

Returns

`MPRGlobalTestFixture` object

Definition at line 206 of file `shared_memory_reader_t.cc`.

7.78.2.4 `art::ProductRegistryHelper& ShmRTestFixture::helper() [inline]`

Get an `art::ProductRegistryHelper`, creating a static instance if necessary.

Returns

`art::ProductRegistryHelper` object

Definition at line 216 of file `shared_memory_reader_t.cc`.

7.78.2.5 `artdaq::detail::SharedMemoryReader& ShmRTestFixture::reader() [inline]`

Get an `artdaq::detail::SharedMemoryReader` object, creating a static instance if necessary.

Returns

`artdaq::detail::SharedMemoryReader` object

Definition at line 272 of file `shared_memory_reader_t.cc`.

7.78.2.6 `art::SourceHelper& ShmRTestFixture::source_helper() [inline]`

Get an `art::SourceHelper` object, creating a static instance if necessary.

Returns

`art::SourceHelper` object

Definition at line 226 of file `shared_memory_reader_t.cc`.

7.78.2.7 `artdaq::SharedMemoryEventManager& ShmRTestFixture::writer() [inline]`

Get the instance of the `SharedMemoryEventManager`.

Returns

Static instance of the `SharedMemoryEventManager`

Definition at line 299 of file `shared_memory_reader_t.cc`.

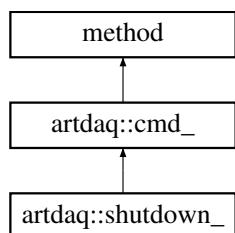
The documentation for this struct was generated from the following file:

- `artdaq/test/ArtModules/shared_memory_reader_t.cc`

7.79 `artdaq::shutdown_` Class Reference

`shutdown_` Command class

Inheritance diagram for `artdaq::shutdown_`:



Public Member Functions

- `shutdown_(xmlrpc_commander &c)`
shutdown_ Constructor

Static Public Attributes

- `static const uint64_t defaultTimeout = 45`

Additional Inherited Members

7.79.1 Detailed Description

[shutdown_](#) Command class

Definition at line 485 of file `xmlrpc_commander.cc`.

7.79.2 Constructor & Destructor Documentation

7.79.2.1 artdaq::shutdown_::shutdown_(`xmlrpc_commander & c`) [inline]

[shutdown_](#) Constructor

Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send transition commands to
----------------	--

Definition at line 492 of file `xmlrpc_commander.cc`.

7.79.3 Member Data Documentation

7.79.3.1 `const uint64_t artdaq::shutdown_::defaultTimeout = 45` [static]

Default timeout for command

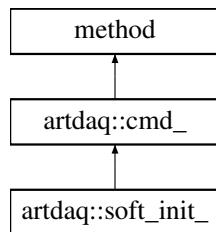
Definition at line 497 of file `xmlrpc_commander.cc`.

The documentation for this class was generated from the following file:

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

7.80 artdaq::soft_init_ Class Reference

Inheritance diagram for `artdaq::soft_init_`:



Public Member Functions

- [soft_init_\(xmlrpc_commander &c\)](#)

Static Public Attributes

- static const `uint64_t defaultTimeout = 45`

- static const uint64_t **defaultTimestamp** = std::numeric_limits<const uint64_t>::max()

Additional Inherited Members

7.80.1 Detailed Description

Command class representing an init transition

Definition at line 391 of file xmlrpc_commander.cc.

7.80.2 Constructor & Destructor Documentation

7.80.2.1 artdaq::soft_init_::soft_init_(xmlrpc_commander & c) [inline], [explicit]

Command class Constructor *

Parameters

c	xmlrpc_commander to send parsed command to \
---	--

Definition at line 391 of file xmlrpc_commander.cc.

7.80.3 Member Data Documentation

7.80.3.1 const uint64_t artdaq::soft_init_::defaultTimeout = 45 [static]

Default timeout for command

Definition at line 391 of file xmlrpc_commander.cc.

7.80.3.2 const uint64_t artdaq::soft_init_::defaultTimestamp = std::numeric_limits<const uint64_t>::max() [static]

Default timestamp for Command

Definition at line 391 of file xmlrpc_commander.cc.

The documentation for this class was generated from the following file:

- artdaq/artdaq/ExternalComms/xmlrpc_commander.cc

7.81 art::Source_generator< artdaq::detail::SharedMemoryReader<> > Struct Template Reference

Specialize an art source trait to tell art that we don't care about source.fileNames and don't want the files services to be used.

Static Public Attributes

- static constexpr bool **value** = true

Used to suppress use of file services on art Source.

7.81.1 Detailed Description

```
template<>struct art::Source_generator< artdaq::detail::SharedMemoryReader<> >
```

Specialize an art source trait to tell art that we don't care about source.fileNames and don't want the files services to be used.

Definition at line 24 of file RawInput_source.cc.

The documentation for this struct was generated from the following file:

- artdaq/artdaq/ArtModules/RawInput_source.cc

7.82 art::Source_generator< ArtdaqInput< artdaq::TransferWrapper >> Struct Template Reference

Trait definition (must precede source typedef).

Static Public Attributes

- static constexpr bool **value** = true

Dummy variable.

7.82.1 Detailed Description

```
template<>struct art::Source_generator< ArtdaqInput< artdaq::TransferWrapper >>
```

Trait definition (must precede source typedef).

Definition at line 10 of file TransferInput_source.cc.

The documentation for this struct was generated from the following file:

- artdaq/artdaq/ArtModules/TransferInput_source.cc

7.83 art::Source_generator< ArtdaqInput< NetMonWrapper >> Struct Template Reference

Trait definition (must precede source typedef).

Static Public Attributes

- static constexpr bool **value** = true

dummy parameter

7.83.1 Detailed Description

```
template<>struct art::Source_generator< ArtdaqInput< NetMonWrapper > >
```

Trait definition (must precede source typedef).

Definition at line 13 of file NetMonInput_source.cc.

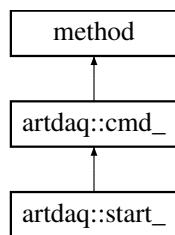
The documentation for this struct was generated from the following file:

- artdaq/artdaq/ArtModules/NetMonInput_source.cc

7.84 artdaq::start_ Class Reference

Command class representing a start transition.

Inheritance diagram for artdaq::start_:



Public Member Functions

- [start_\(xmlrpc_commander &c\)](#)
start_ Command (cmd_ derived class) Constructor

Static Public Attributes

- static const uint64_t [defaultTimeout](#) = 45
- static const uint64_t [defaultTimestamp](#) = std::numeric_limits<const uint64_t>::max()

Additional Inherited Members

7.84.1 Detailed Description

Command class representing a start transition.

Definition at line 402 of file xmlrpc_commander.cc.

7.84.2 Constructor & Destructor Documentation

7.84.2.1 artdaq::start_::start_(xmlrpc_commander & c) [inline], [explicit]

[start_ Command \(cmd_ derived class\) Constructor](#)

Parameters

c	xmlrpc_commander instance to command
---	--

Definition at line 409 of file [xmlrpc_commander.cc](#).

7.84.3 Member Data Documentation

7.84.3.1 const uint64_t artdaq::start_::defaultTimeout = 45 [static]

Default timeout for command

Definition at line 414 of file [xmlrpc_commander.cc](#).

7.84.3.2 const uint64_t artdaq::start_::defaultTimestamp = std::numeric_limits<const uint64_t>::max() [static]

Default timestamp for Command

Definition at line 416 of file [xmlrpc_commander.cc](#).

The documentation for this class was generated from the following file:

- [artdaq/artdaq/ExternalComms/xmlrpc_commander.cc](#)

7.85 artdaq::StatisticsHelper Class Reference

This class manages MonitoredQuantity instances for the *Core classes.

```
#include <artdaq/Application/StatisticsHelper.hh>
```

Public Member Functions

- [StatisticsHelper \(\)](#)
StatisticsHelper default constructor.
- [StatisticsHelper \(StatisticsHelper const &\)=delete](#)
Copy Constructor is deleted.
- [virtual ~StatisticsHelper \(\)=default](#)
Default Destructor.
- [StatisticsHelper & operator= \(StatisticsHelper const &\)=delete](#)
Copy Assignment operator is deleted.
- [void addMonitoredQuantityName \(std::string const &statKey\)](#)
Add a MonitoredQuantity name to the list.
- [void addSample \(std::string const &statKey, double value\) const](#)
Add a sample to the MonitoredQuantity with the given name.
- [bool createCollectors \(fhicl::ParameterSet const &pset, int defaultReportIntervalFragments, double defaultReportIntervalSeconds, double defaultMonitorWindow, std::string const &primaryStatKeyName\)](#)
Create MonitoredQuantity objects for all names registered with the [StatisticsHelper](#).
- [void resetStatistics \(\)](#)
Reset all MonitoredQuantity instances.
- [bool readyToReport \(size_t currentCount\)](#)

Determine if the reporting interval conditions have been met.

- bool [statsRollingWindowHasMoved \(\)](#)

Determine if the MonitoredQuantity "recent" window has changed since the last time this function was called.

7.85.1 Detailed Description

This class manages MonitoredQuantity instances for the *Core classes.

Definition at line 16 of file StatisticsHelper.hh.

7.85.2 Member Function Documentation

7.85.2.1 void artdaq::StatisticsHelper::addMonitoredQuantityName (std::string const & statKey)

Add a MonitoredQuantity name to the list.

Parameters

<code>statKey</code>	Name of the MonitoredQuantity to be added
----------------------	---

Definition at line 15 of file StatisticsHelper.cc.

7.85.2.2 void artdaq::StatisticsHelper::addSample (std::string const & statKey, double value) const

Add a sample to the MonitoredQuantity with the given name.

Parameters

<code>statKey</code>	Name of the MonitoredQuantity
<code>value</code>	Value to record in the MonitoredQuantity

Definition at line 20 of file StatisticsHelper.cc.

7.85.2.3 bool artdaq::StatisticsHelper::createCollectors (fhicl::ParameterSet const & pset, int defaultReportIntervalFragments, double defaultReportIntervalSeconds, double defaultMonitorWindow, std::string const & primaryStatKeyName)

Create MonitoredQuantity objects for all names registered with the [StatisticsHelper](#).

Parameters

<code>pset</code>	ParameterSet used to configure reporting
<code>defaultReport-Interval-Fragments</code>	Default reporting interval in Fragments
<code>defaultReport-IntervalSeconds</code>	Default reporting interval in Seconds
<code>defaultMonitor-Window</code>	Default monitoring window

<code>primaryStatKey- Name</code>	The primary (default) MonitoredQuantity
---------------------------------------	---

Returns

Whether the primary MonitoredQuantity exists

StatisticsHelper accepts the following Parameters: "reporting_interval.fragments" (Default given above): The reporting interval in Fragments "reporting_interval.seconds" (Default given above): The reporting interval in Seconds "monitor_window" (Default given above): The monitoring window for the MonitoredQuantity "monitor_binsize" (Default: 1 + ((monitorWindow - 1) / 100)): The monitoring bin size for the MonitoredQuantity

Definition at line 29 of file StatisticsHelper.cc.

7.85.2.4 StatisticsHelper& artdaq::StatisticsHelper::operator= (StatisticsHelper const &) [delete]

Copy Assignment operator is deleted.

Returns

[StatisticsHelper](#) copy

7.85.2.5 bool artdaq::StatisticsHelper::readyToReport (size_t currentCount)

Determine if the reporting interval conditions have been met.

Parameters

<code>currentCount</code>	Current Fragment count
---------------------------	------------------------

Returns

Whether the [StatisticsHelper](#) is ready to report

Definition at line 74 of file StatisticsHelper.cc.

7.85.2.6 bool artdaq::StatisticsHelper::statsRollingWindowHasMoved ()

Determine if the MonitoredQuantity "recent" window has changed since the last time this function was called.

Returns

Whether the MonitoredQuantity "recent" window has changed

Definition at line 91 of file StatisticsHelper.cc.

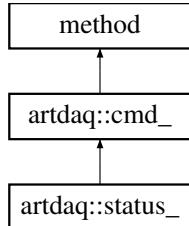
The documentation for this class was generated from the following files:

- artdaq/artdaq/Application/StatisticsHelper.hh
- artdaq/artdaq/Application/StatisticsHelper.cc

7.86 artdaq::status_ Class Reference

[status_ Command class](#)

Inheritance diagram for artdaq::status_:



Public Member Functions

- [status_ \(xmlrpc_commander &c\)](#)

status_ Constructor

Additional Inherited Members

7.86.1 Detailed Description

[status_ Command class](#)

Definition at line 511 of file `xmlrpc_commander.cc`.

7.86.2 Constructor & Destructor Documentation

7.86.2.1 artdaq::status_::status_ (`xmlrpc_commander & c`) [inline]

[status_ Constructor](#)

Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send transition commands to
----------------	--

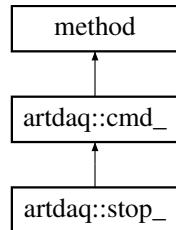
Definition at line 518 of file `xmlrpc_commander.cc`.

The documentation for this class was generated from the following file:

- artdaq/artdaq/ExternalComms/xmlrpc_commander.cc

7.87 artdaq::stop_ Class Reference

Inheritance diagram for artdaq::stop_:



Public Member Functions

- [stop_ \(xmlrpc_commander &c\)](#)

Static Public Attributes

- static const uint64_t `defaultTimeout` = 45
- static const uint64_t `defaultTimestamp` = std::numeric_limits<const uint64_t>::max()

Additional Inherited Members

7.87.1 Detailed Description

[stop_ Command class](#)

Definition at line 477 of file `xmlrpc_commander.cc`.

7.87.2 Constructor & Destructor Documentation

7.87.2.1 artdaq::stop_::stop_ (`xmlrpc_commander & c`) [inline]

[stop_ Constructor \](#)

Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send transition commands to
----------------	--

Definition at line 477 of file `xmlrpc_commander.cc`.

7.87.3 Member Data Documentation

7.87.3.1 `const uint64_t artdaq::stop_::defaultTimeout = 45` [static]

Default timeout for command

Definition at line 477 of file `xmlrpc_commander.cc`.

7.87.3.2 `const uint64_t artdaq::stop_::defaultTimestamp = std::numeric_limits<const uint64_t>::max()` [static]

Default timestamp for Command

Definition at line 477 of file `xmlrpc_commander.cc`.

The documentation for this class was generated from the following file:

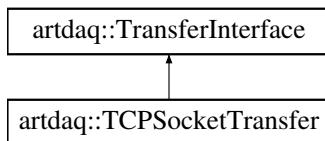
- artdaq/artdaq/ExternalComms/xmlrpc_commander.cc

7.88 artdaq::TCPSocketTransfer Class Reference

[TransferInterface](#) implementation plugin that sends data using TCP sockets.

```
#include <artdaq/TransferPlugins/TCPSocketTransfer.hh>
```

Inheritance diagram for artdaq::TCPSocketTransfer:



Public Member Functions

- [TCPSocketTransfer](#) (fhicl::ParameterSet const &ps, [Role role](#))
TCPSocketTransfer Constructor.
- int [receiveFragmentHeader](#) (detail::RawFragmentHeader &header, size_t receiveTimeout) override
Receive a Fragment Header from the transport mechanism.
- int [receiveFragmentData](#) (RawDataType *destination, size_t wordCount) override
Receive the body of a Fragment to the given destination pointer.
- [CopyStatus copyFragment](#) (Fragment &frag, size_t timeout_usec) override
Copy a Fragment to the destination. Same implementation as moveFragment, as TCP is always reliable.
- [CopyStatus moveFragment](#) (Fragment &&frag, size_t timeout_usec) override
Move a Fragment to the destination.

Additional Inherited Members

7.88.1 Detailed Description

[TransferInterface](#) implementation plugin that sends data using TCP sockets.

Definition at line 35 of file TCPSocketTransfer.hh.

7.88.2 Constructor & Destructor Documentation

7.88.2.1 artdaq::TCPSocketTransfer::TCPSocketTransfer (fhicl::ParameterSet const & ps, [TransferInterface::Role role](#))

[TCPSocketTransfer](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure TCPSocketTransfer
<i>role</i>	Role of this TCPSocketTransfer instance (kSend or kReceive)

* TCPSocketTransfer accepts the following Parameters:
 * "tcp_receive_buffer_size" (Default: 0): The TCP buffer size on the receive socket
 * "host_map" (REQUIRED): List of FHiCL tables containing information about other hosts in the system.
 * Each table should contain:
 * "rank" (Default: RECV_TIMEOUT): Rank of this host
 * "host" (Default: "localhost"): Hostname of this host
 * "portOffset" (Default: 5500): To avoid collisions, each destination should specify its own port offset.
 * All TCPSocketTransfers sending to that destination will add their own rank to make a unique port number.
 *

[TCPSocketTransfer](#) also requires all Parameters for configuring a [TransferInterface](#)

Definition at line 35 of file TCPSocket_transfer.cc.

7.88.3 Member Function Documentation

7.88.3.1 [CopyStatus artdaq::TCPSocketTransfer::copyFragment \(Fragment & frag, size_t timeout_usec \) \[inline\], \[override\]](#)

Copy a Fragment to the destination. Same implementation as moveFragment, as TCP is always reliable.

Parameters

<i>frag</i>	Fragment to copy
<i>timeout_usec</i>	Timeout for send, in microseconds

Returns

[CopyStatus](#) detailing result of copy

Definition at line 81 of file TCPSocketTransfer.hh.

7.88.3.2 [CopyStatus artdaq::TCPSocketTransfer::moveFragment \(Fragment && frag, size_t timeout_usec \) \[inline\], \[override\]](#)

Move a Fragment to the destination.

Parameters

<i>frag</i>	Fragment to move
<i>timeout_usec</i>	Timeout for send, in microseconds

Returns

[CopyStatus](#) detailing result of copy

Definition at line 89 of file TCPSocketTransfer.hh.

7.88.3.3 [int artdaq::TCPSocketTransfer::receiveFragmentData \(RawDataType * destination, size_t wordCount \) \[override\], \[virtual\]](#)

Receive the body of a Fragment to the given destination pointer.

Parameters

<i>destination</i>	Pointer to memory region where Fragment data should be stored
<i>wordCount</i>	Number of RawDataType words to receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implements [artdaq::TransferInterface](#).

Definition at line 217 of file TCPSocket_transfer.cc.

```
7.88.3.4 int artdaq::TCPSocketTransfer::receiveFragmentHeader ( detail::RawFragmentHeader & header, size_t receiveTimeout )
[override], [virtual]
```

Receive a Fragment Header from the transport mechanism.

Parameters

<i>out</i>	<i>header</i>	Received Fragment Header
	<i>receiveTimeout</i>	Timeout for receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implements [artdaq::TransferInterface](#).

Definition at line 103 of file TCPSocket_transfer.cc.

The documentation for this class was generated from the following files:

- [artdaq/artdaq/TransferPlugins/TCPSocketTransfer.hh](#)
- [artdaq/artdaq/TransferPlugins/TCPSocket_transfer.cc](#)

7.89 anonymous_namespace{genToArt.cc}::ThrottledGenerator Class Reference

[ThrottledGenerator](#): ensure that we only get one fragment per type at a time from the generator.

Public Member Functions

- **ThrottledGenerator** (std::string const &generator, fhicl::ParameterSet const &ps)

ThrottledGenerator Constructor.
- bool **getNext** (artdaq::FragmentPtrs &newFrags)

Get the next fragment from the generator.
- size_t **numFragIDs** () const

Get the number of Fragment IDs handled by this generator.
- void **start** (int run, uint64_t timeout, uint64_t timestamp) const

Send start signal to FragmentGenerator, if it's a CommandableFragmentGenerator.
- void **stop** (uint64_t timeout, uint64_t timestamp) const

Send stop signal to FragmentGenerator, if it's a CommandableFragmentGenerator.

7.89.1 Detailed Description

[ThrottledGenerator](#): ensure that we only get one fragment per type at a time from the generator.

Definition at line 89 of file genToArt.cc.

7.89.2 Constructor & Destructor Documentation

7.89.2.1 `anonymous_namespace{genToArt.cc}::ThrottledGenerator (std::string const & generator, fhicl::ParameterSet const & ps)`

[ThrottledGenerator](#) Constructor.

Parameters

<code>generator</code>	Name of the generator plugin to load
<code>ps</code>	ParameterSet for configuring the FragmentGenerator

Definition at line 145 of file genToArt.cc.

7.89.3 Member Function Documentation

7.89.3.1 `bool anonymous_namespace{genToArt.cc}::ThrottledGenerator::getNext (artdaq::FragmentPtrs & newFrags)`

Get the next fragment from the generator.

Parameters

<code>out</code>	<code>newFrags</code>	New Fragment objects are added to this list
------------------	-----------------------	---

Returns

Whether there is more data forthcoming

Definition at line 157 of file genToArt.cc.

7.89.3.2 `size_t anonymous_namespace{genToArt.cc}::ThrottledGenerator::numFragIDs () const`

Get the number of Fragment IDs handled by this generator.

Returns

Definition at line 193 of file genToArt.cc.

7.89.3.3 `void anonymous_namespace{genToArt.cc}::ThrottledGenerator::start (int run, uint64_t timeout, uint64_t timestamp) const [inline]`

Send start signal to FragmentGenerator, if it's a CommandableFragmentGenerator.

Parameters

<i>run</i>	Run number to pass to StartCmd
<i>timeout</i>	Timeout to pass to StartCmd
<i>timestamp</i>	Timestamp to pass to StartCmd

Definition at line 119 of file genToArt.cc.

7.89.3.4 void anonymous_namespace{genToArt.cc}::ThrottledGenerator::stop (uint64_t *timeout*, uint64_t *timestamp*) const [inline]

Send stop signal to FragmentGenerator, if it's a CommandableFragmentGenerator.

Parameters

<i>timeout</i>	Timeout to pass to StopCmd
<i>timestamp</i>	Timestamp to pass to StopCmd

Definition at line 129 of file genToArt.cc.

The documentation for this class was generated from the following file:

- artdaq/tools/genToArt.cc

7.90 Timeout Class Reference

The [Timeout](#) class performs registered actions at specified intervals.

```
#include <artdaq/TransferPlugins/detail/Timeout.hh>
```

Classes

- struct [timeoutspec](#)

Specification for a [Timeout](#) function.

Public Member Functions

- [Timeout](#) (int max_tmos=100)
Construct a [Timeout](#) object.
- void [add_periodic](#) (const char *desc, void *tag, std::function< void()> &function, uint64_t period_us, uint64_t start_us=0)
Add a periodic timeout to the [Timeout](#) container.
- void [add_periodic](#) (const char *desc, void *tag, std::function< void()> &function, int rel_ms)
Add a periodic timeout to the [Timeout](#) container.
- void [add_periodic](#) (const char *desc, uint64_t period_us, uint64_t start_us=0)
Add a periodic timeout to the [Timeout](#) container.
- void [add_relative](#) (const char *desc, void *tag, std::function< void()> &function, int rel_ms)
Add a periodic timeout to the [Timeout](#) container.
- void [add_relative](#) (std::string desc, int rel_ms)
Add a periodic timeout to the [Timeout](#) container.
- void [copy_in_timeout](#) (const char *desc, uint64_t period_us, uint64_t start_us=0)

Add a timeout with the given parameters.

- bool `cancel_timeout` (void *tag, std::string desc)

Cancel the timeout having the given description and tag.

- int `get_next_expired_timeout` (std::string &desc, void **tag, std::function< void()> &function, uint64_t *tmo_us)

Get a timeout that has expired.

- void `get_next_timeout_delay` (int64_t *delay_us)

Get the amount to wait for the next timeout to occur.

- int `get_next_timeout_msdl` ()

Get the amount to wait for the next timeout to occur.

- bool `is_consistent` ()

Run a consistency check on all configured timeouts.

- void `list_active_time` ()

TRACE all active timeouts.

7.90.1 Detailed Description

The `Timeout` class performs registered actions at specified intervals.

Definition at line 22 of file `Timeout.hh`.

7.90.2 Constructor & Destructor Documentation

7.90.2.1 `Timeout::Timeout(int max_tmos = 100) [explicit]`

Construct a `Timeout` object.

Parameters

<code>max_tmos</code>	Maximum number of registered <code>Timeout</code> functions
-----------------------	---

Definition at line 52 of file `Timeout.cc`.

7.90.3 Member Function Documentation

7.90.3.1 `void Timeout::add_periodic(const char * desc, void * tag, std::function< void()> & function, uint64_t period_us, uint64_t start_us = 0)`

Add a periodic timeout to the `Timeout` container.

Parameters

<code>desc</code>	Description of the periodic timeout
<code>tag</code>	Tag (fd or other) to apply to timeout
<code>function</code>	Function to execute at timeout
<code>period_us</code>	Period for timeouts
<code>start_us</code>	When to start (defaults to 0)

maybe need to return a timeout id??

Definition at line 61 of file `Timeout.cc`.

7.90.3.2 void Timeout::add_periodic (const char * *desc*, void * *tag*, std::function< void()> & *function*, int *rel_ms*)

Add a periodic timeout to the [Timeout](#) container.

Parameters

<i>desc</i>	Description of the periodic timeout
<i>tag</i>	Tag (fd or other) to apply to timeout
<i>function</i>	Function to execute at timeout
<i>rel_ms</i>	Timeout in rem_ms milliseconds

maybe need to return a timeout id??

Definition at line 76 of file Timeout.cc.

7.90.3.3 void Timeout::add_periodic (const char * *desc*, uint64_t *period_us*, uint64_t *start_us* = 0)

Add a periodic timeout to the [Timeout](#) container.

Parameters

<i>desc</i>	Description of the periodic timeout
<i>period_us</i>	Period for timeouts
<i>start_us</i>	When to start (defaults to 0)

maybe need to return a timeout id??

Definition at line 90 of file Timeout.cc.

7.90.3.4 void Timeout::add_relative (const char * *desc*, void * *tag*, std::function< void() > & *function*, int *rel_ms*)

Add a periodic timeout to the [Timeout](#) container.

Parameters

<i>desc</i>	Description of the periodic timeout
<i>tag</i>	Tag (fd or other) to apply to timeout
<i>function</i>	Function to execute at timeout
<i>rel_ms</i>	Timeout in rem_ms milliseconds

maybe need to return a timeout id??

Definition at line 107 of file Timeout.cc.

7.90.3.5 void Timeout::add_relative (std::string *desc*, int *rel_ms*)

Add a periodic timeout to the [Timeout](#) container.

Parameters

<i>desc</i>	Description of the periodic timeout
<i>rel_ms</i>	Timeout in rem_ms milliseconds

maybe need to return a timeout id??

Definition at line 121 of file Timeout.cc.

7.90.3.6 bool Timeout::cancel_timeout (void * *tag*, std::string *desc*)

Cancel the timeout having the given description and tag.

Parameters

<i>tag</i>	Tag of the cancelled timeout
<i>desc</i>	Description of the cancelled timeout

Returns

Whether a timeout was found and cancelled

Definition at line 321 of file Timeout.cc.

7.90.3.7 void Timeout::copy_in_timeout (const char * *desc*, uint64_t *period_us*, uint64_t *start_us* = 0)

Add a timeout with the given parameters.

Parameters

<i>desc</i>	Description of new timeout
<i>period_us</i>	Period that the timeout should execute with
<i>start_us</i>	When to start the timeout (default 0)

Definition at line 291 of file Timeout.cc.

7.90.3.8 int Timeout::get_next_expired_timeout (std::string & *desc*, void ** *tag*, std::function< void()> & *function*, uint64_t * *tmo_tod_us*)

Get a timeout that has expired.

Parameters

<i>out</i>	<i>desc</i>	Description of timeout that expired
<i>out</i>	<i>tag</i>	Tag of timeout that expired
<i>out</i>	<i>function</i>	Function of timeout that expired
<i>out</i>	<i>tmo_tod_us</i>	When the timeout expired

Returns

-1 if no timeouts expired

Definition at line 135 of file Timeout.cc.

7.90.3.9 void Timeout::get_next_timeout_delay (int64_t * *delay_us*)

Get the amount to wait for the next timeout to occur.

Parameters

<i>out</i>	<i>delay_us</i>	Microseconds until next timeout
------------	-----------------	---------------------------------

Definition at line 158 of file Timeout.cc.

7.90.3.10 int Timeout::get_next_timeout_msdlly ()

Get the amount to wait for the next timeout to occur.

Returns

Milliseconds until next timeout

Definition at line 179 of file Timeout.cc.

7.90.3.11 bool Timeout::is_consistent()

Run a consistency check on all configured timeouts.

Returns

True if all timeouts pass check

Definition at line 197 of file Timeout.cc.

The documentation for this class was generated from the following files:

- artdaq/artdaq/TransferPlugins/detail/Timeout.hh
- artdaq/artdaq/TransferPlugins/detail/Timeout.cc

7.91 Timeout::timeoutspec Struct Reference

Specification for a [Timeout](#) function.

```
#include <artdaq/TransferPlugins/detail/Timeout.hh>
```

Public Attributes

- std::string **desc**
Description of the [Timeout](#) function.
- void * **tag**
could be file descriptor (fd)
- std::function< void()> **function**
Function to execute at [Timeout](#).
- uint64_t **tmo_tod_us**
When the function should be executed (gettimeofday, microseconds)
- uint64_t **period_us**
0 if not periodic
- int **missed_periods**
Number of periods that passed while the function was executing.
- int **check**
Check the timeoutspec.

7.91.1 Detailed Description

Specification for a [Timeout](#) function.

Definition at line 28 of file Timeout.hh.

The documentation for this struct was generated from the following file:

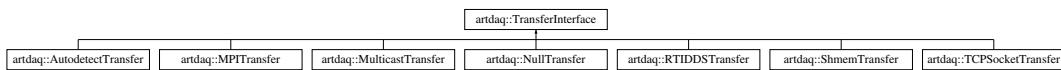
- artdaq/artdaq/TransferPlugins/detail/Timeout.hh

7.92 artdaq::TransferInterface Class Reference

This interface defines the functions used to transfer data between artdaq applications.

```
#include <artdaq/TransferPlugins/TransferInterface.hh>
```

Inheritance diagram for artdaq::TransferInterface:



Public Types

- enum `Role` { `Role::kSend`, `Role::kReceive` }

Used to determine if a `TransferInterface` is a Sender or Receiver.

- enum `CopyStatus` { `CopyStatus::kSuccess`, `CopyStatus::kTimeout`, `CopyStatus::kErrorNotRequiringException` }

Returned from the send functions, this enumeration describes the possible return codes. If an exception occurs, it will be thrown and should be handled normally.

Public Member Functions

- `TransferInterface (const fhicl::ParameterSet &ps, Role role)`
TransferInterface Constructor.
- `TransferInterface (const TransferInterface &)=delete`
Copy Constructor is deleted.
- `TransferInterface & operator= (const TransferInterface &)=delete`
Copy Assignment operator is deleted.
- `virtual ~TransferInterface ()=default`
Default virtual Destructor.
- `virtual int receiveFragment (artdaq::Fragment &fragment, size_t receiveTimeout)`
Receive a Fragment from the transport mechanism.
- `virtual int receiveFragmentHeader (detail::RawFragmentHeader &header, size_t receiveTimeout)=0`
Receive a Fragment Header from the transport mechanism.
- `virtual int receiveFragmentData (RawDataType *destination, size_t wordCount)=0`
Receive the body of a Fragment to the given destination pointer.
- `virtual CopyStatus copyFragment (artdaq::Fragment &fragment, size_t send_timeout_usec=std::numeric_limits<size_t>::max())=0`
Copy a Fragment to the destination. May not necessarily be reliable.
- `virtual CopyStatus moveFragment (artdaq::Fragment &&fragment, size_t send_timeout_usec=std::numeric_limits<size_t>::max())=0`
Move a Fragment to the destination. This should be reliable, if the underlying transport mechanism supports reliable sending.
- `std::string uniqueLabel () const`
Get the unique label of this `TransferInterface` instance.
- `virtual int source_rank () const`
Get the source rank for this `TransferInterface` instance.
- `virtual int destination_rank () const`
Get the destination rank for this `TransferInterface` instance.

Static Public Attributes

- static const int **RECV_TIMEOUT** = 0xfedcba98

Value to be returned upon receive timeout. Because receivers otherwise return rank, this is also the limit on the number of ranks that artdaq currently supports.

Protected Member Functions

- **Role role () const**
Get the TransferInterface::Role of this TransferInterface.

Protected Attributes

- size_t **buffer_count_**
The number of Fragment transfers the TransferInterface can handle simultaneously.
- const size_t **max_fragment_size_words_**
The maximum size of the transferred Fragment objects, in artdaq::Fragment::RawDataType words.

7.92.1 Detailed Description

This interface defines the functions used to transfer data between artdaq applications.

Definition at line 17 of file TransferInterface.hh.

7.92.2 Member Enumeration Documentation

7.92.2.1 enum artdaq::TransferInterface::CopyStatus [strong]

Returned from the send functions, this enumeration describes the possible return codes. If an exception occurs, it will be thrown and should be handled normally.

Enumerator

- kSuccess** The send operation completed successfully.
- kTimeout** The send operation timed out.
- kErrorNotRequiringException** Some error occurred, but no exception was thrown.

Definition at line 39 of file TransferInterface.hh.

7.92.2.2 enum artdaq::TransferInterface::Role [strong]

Used to determine if a TransferInterface is a Sender or Receiver.

Enumerator

- kSend** This TransferInterface is a Sender.
- kReceive** This TransferInterface is a Receiver.

Definition at line 29 of file TransferInterface.hh.

7.92.3 Constructor & Destructor Documentation

7.92.3.1 artdaq::TransferInterface::TransferInterface (const fhicl::ParameterSet & ps, Role role)

[TransferInterface](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used for configuring the TransferInterface
<i>role</i>	Role of the TransferInterface (See TransferInterface::Role)

```
* TransferInterface accepts the following Parameters:
* "source_rank" (Default: my_rank): The rank that data is coming from
* "destination_rank" (Default: my_rank): The rank that data is going to
* "unique_label" (Default: "transfer_between_[source_rank]_and_[destination_rank]"): A label that uniquely identifies the transfer
* "buffer_count" (Default: 10): How many Fragments can the TransferInterface handle simultaneously
* "max_fragment_size_words" (Default: 1024): The maximum Fragment size expected. May be used for static memory allocation if larger Fragments are sent.
*
```

Definition at line 4 of file TransferInterface.cc.

7.92.4 Member Function Documentation

7.92.4.1 virtual CopyStatus artdaq::TransferInterface::copyFragment (artdaq::Fragment & fragment, size_t send_timeout_usec = std::numeric_limits< size_t >::max()) [pure virtual]

Copy a Fragment to the destination. May not necessarily be reliable.

Parameters

<i>fragment</i>	Fragment to copy
<i>send_timeout_usec</i>	Timeout for send, in microseconds

Returns

CopyStatus detailing result of copy

Implemented in [artdaq::MulticastTransfer](#), [artdaq::AutodetectTransfer](#), [artdaq::NullTransfer](#), and [artdaq::RTIDDS-Transfer](#).

7.92.4.2 virtual int artdaq::TransferInterface::destination_rank () const [inline], [virtual]

Get the destination rank for this [TransferInterface](#) instance.

Returns

The destination rank for this [TransferInterface](#) instance

Definition at line 138 of file TransferInterface.hh.

7.92.4.3 virtual CopyStatus artdaq::TransferInterface::moveFragment (artdaq::Fragment && fragment, size_t send_timeout_usec = std::numeric_limits< size_t >::max()) [pure virtual]

Move a Fragment to the destination. This should be reliable, if the underlying transport mechanism supports reliable sending.

Parameters

<i>fragment</i>	Fragment to move
<i>send_timeout_-usec</i>	Timeout for send, in microseconds

Returns

CopyStatus detailing result of copy

Implemented in [ardaq::MulticastTransfer](#), [ardaq::AutodetectTransfer](#), [ardaq::NullTransfer](#), and [ardaq::RTIDDS-Transfer](#).

7.92.4.4 TransferInterface& artdaq::TransferInterface::operator=(const TransferInterface &) [delete]

Copy Assignment operator is deleted.

Returns

[TransferInterface](#) Copy

7.92.4.5 int artdaq::TransferInterface::receiveFragment(artdaq::Fragment & fragment, size_t receiveTimeout) [virtual]

Receive a Fragment from the transport mechanism.

Parameters

<i>out</i>	<i>fragment</i>	Received Fragment
	<i>receiveTimeout</i>	Timeout for receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Reimplemented in [ardaq::MulticastTransfer](#), [ardaq::RTIDDSTransfer](#), [ardaq::AutodetectTransfer](#), and [ardaq::NullTransfer](#).

Definition at line 15 of file TransferInterface.cc.

7.92.4.6 virtual int artdaq::TransferInterface::receiveFragmentData(RawDataType * destination, size_t wordCount) [pure virtual]

Receive the body of a Fragment to the given destination pointer.

Parameters

<i>destination</i>	Pointer to memory region where Fragment data should be stored
<i>wordCount</i>	Number of words of Fragment data to receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

The precondition for calling this function is that you have received a valid header, therefore it does not have a , as the Fragment data should immediately be available.

Implemented in [artdaq::MPITransfer](#), [artdaq::MulticastTransfer](#), [artdaq::TCPSocketTransfer](#), [artdaq::ShmemTransfer](#), [artdaq::AutodetectTransfer](#), and [artdaq::NullTransfer](#).

7.92.4.7 virtual int artdaq::TransferInterface::receiveFragmentHeader (detail::RawFragmentHeader & header, size_t receiveTimeout) [pure virtual]

Receive a Fragment Header from the transport mechanism.

Parameters

<i>out</i>	<i>header</i>	Received Fragment Header
	<i>receiveTimeout</i>	Timeout for receive

Returns

The rank the Fragment was received from (should be source_rank), or RECV_TIMEOUT

Implemented in [artdaq::MPITransfer](#), [artdaq::MulticastTransfer](#), [artdaq::TCPSocketTransfer](#), [artdaq::ShmemTransfer](#), [artdaq::AutodetectTransfer](#), and [artdaq::NullTransfer](#).

7.92.4.8 Role artdaq::TransferInterface::role () const [inline], [protected]

Get the [TransferInterface::Role](#) of this [TransferInterface](#).

Returns

The Role of this [TransferInterface](#)

Definition at line 154 of file TransferInterface.hh.

7.92.4.9 virtual int artdaq::TransferInterface::source_rank () const [inline], [virtual]

Get the source rank for this [TransferInterface](#) instance.

Returns

The source rank for this Transferinterface instance

Definition at line 133 of file TransferInterface.hh.

7.92.4.10 std::string artdaq::TransferInterface::uniqueLabel () const [inline]

Get the unique label of this [TransferInterface](#) instance.

Returns

The unique label of this [TransferInterface](#) instance

Definition at line 127 of file TransferInterface.hh.

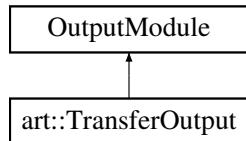
The documentation for this class was generated from the following files:

- artdaq/artdaq/TransferPlugins/TransferInterface.hh
- artdaq/artdaq/TransferPlugins/TransferInterface.cc

7.93 art::TransferOutput Class Reference

An art::OutputModule which sends events using DataSenderManager. This module is designed for transporting Fragment-wrapped art::Events after they have been read into art, for example between the EventBuilder and the Aggregator.

Inheritance diagram for art::TransferOutput:



Public Member Functions

- [TransferOutput \(fhicl::ParameterSet const &ps\)](#)
TransferOutput Constructor.
- [~TransferOutput \(\)](#)
TransferOutput Destructor.

7.93.1 Detailed Description

An art::OutputModule which sends events using DataSenderManager. This module is designed for transporting Fragment-wrapped art::Events after they have been read into art, for example between the EventBuilder and the Aggregator.

Definition at line 67 of file TransferOutput_module.cc.

7.93.2 Constructor & Destructor Documentation

7.93.2.1 art::TransferOutput::TransferOutput (fhicl::ParameterSet const & ps) [explicit]

[TransferOutput](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure TransferOutput
-----------	---

[TransferOutput](#) accepts no Parameters beyond those which art::OutputModule takes. See the art::OutputModule documentation for more details on those Parameters.

Definition at line 114 of file TransferOutput_module.cc.

The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/TransferOutput_module.cc

7.94 artdaq::TransferTest Class Reference

Test a set of [TransferInterface](#) plugins.

```
#include <proto/TransferTest.hh>
```

Public Member Functions

- [TransferTest](#) (fhicl::ParameterSet *psi*)
TransferTest Constructor.
- int [runTest](#) ()
Run the test as configured.

7.94.1 Detailed Description

Test a set of [TransferInterface](#) plugins.

Definition at line 17 of file TransferTest.hh.

7.94.2 Constructor & Destructor Documentation

7.94.2.1 artdaq::TransferTest::TransferTest (fhicl::ParameterSet *psi*) [explicit]

[TransferTest](#) Constructor.

Parameters

<i>psi</i>	ParameterSet used to configure TransferTest
------------	---

- * TransferTest accepts the following Parameters:
- * "num_senders" (REQUIRED): Number of sending TransferTest instances
- * "num_receivers" (REQUIRED): Number of receiving TransferTest instances
- * "sends_pper_sender" (REQUIRED): Number of sends each sender will perform
- * "buffer_count" (Default: 10): Buffer count for TransferInterfaces
- * "fragment_size" (Default: 0x100000): Size of Fragments to transfer
- * "metrics": FHiCL table used to configure MetricManager (see documentation)
- * "transfer_plugin_type" (Default: Shmem): TransferInterface plugin to load
- * "hostmap" (OPTIONAL): Host map to use for "host_map" parameter of TransferInterface plugins (i.e. TCPSocketTrans
- *

Definition at line 11 of file TransferTest.cc.

7.94.3 Member Function Documentation

7.94.3.1 int artdaq::TransferTest::runTest()

Run the test as configured.

Returns

0 upon success

Definition at line 83 of file TransferTest.cc.

The documentation for this class was generated from the following files:

- artdaq/proto/TransferTest.hh
- artdaq/proto/TransferTest.cc

7.95 artdaq::TransferWrapper Class Reference

[TransferWrapper](#) wraps a [TransferInterface](#) so that it can be used in the ArtdaqInput class to make an art::Source.

```
#include <artdaq/ArtModules/detail/TransferWrapper.hh>
```

Public Member Functions

- [TransferWrapper \(const fhicl::ParameterSet &pset\)](#)
TransferWrapper Constructor.
- virtual ~[TransferWrapper \(\)](#)
TransferWrapper Destructor.
- void [receiveMessage \(std::unique_ptr< TBufferFile > &msg\)](#)
Receive a Fragment from the TransferInterface, and send it to art.
- void [receiveInitMessage \(std::unique_ptr< TBufferFile > &msg\)](#)
Receive the Init message from the TransferInterface, and send it to art.

7.95.1 Detailed Description

[TransferWrapper](#) wraps a [TransferInterface](#) so that it can be used in the ArtdaqInput class to make an art::Source.

JCF, May-27-2016

This is the class through which code that wants to access a transfer plugin (e.g., input sources, AggregatorCore, etc.) can do so. Its functionality is such that it satisfies the requirements needed to be a template in the ArtdaqInput class

Definition at line 34 of file TransferWrapper.hh.

7.95.2 Constructor & Destructor Documentation

7.95.2.1 artdaq::TransferWrapper::TransferWrapper (const fhicl::ParameterSet & pset) [explicit]

[TransferWrapper](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure the TransferWrapper
-------------	--

```
* TransferWrapper accepts the following Parameters:
* "timeoutInUsecs" (Default: 100000): The receive timeout
* "dispatcherHost" (REQUIRED): The hostname that the Dispatcher Aggregator is running on
* "dispatcherPort" (REQUIRED): The port that the Dispatcher Aggregator is running on
* "maxEventsBeforeInit" (Default: 5): How many non-Init events to receive before raising an error
* "allowedFragmentTypes" (Default: [226,227,229]): The Fragment type codes for expected Fragments
* "quitOnFragmentIntegrityProblem" (Default: true): If there is an inconsistency in the received Fragment, throw an error
* "debugLevel" (Default: 0): Enables some additional messages
* "transfer_plugin" (REQUIRED): Name of the TransferInterface plugin to load
*
* This parameter set is also passed to TransferInterface, so any necessary Parameters for TransferInterface or the TransferInterface plugin should be included here.
```

Definition at line 35 of file TransferWrapper.cc.

7.95.3 Member Function Documentation

7.95.3.1 void artdaq::TransferWrapper::receiveInitMessage (std::unique_ptr< TBufferFile > & msg) [inline]

Receive the Init message from the [TransferInterface](#), and send it to art.

Parameters

<i>out</i>	<i>msg</i>	The message in art format
------------	------------	---------------------------

Definition at line 74 of file TransferWrapper.hh.

7.95.3.2 void artdaq::TransferWrapper::receiveMessage (std::unique_ptr< TBufferFile > & msg)

Receive a Fragment from the [TransferInterface](#), and send it to art.

Parameters

<i>out</i>	<i>msg</i>	The message in art format
------------	------------	---------------------------

Definition at line 82 of file TransferWrapper.cc.

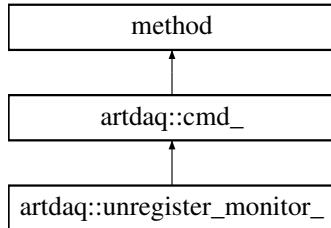
The documentation for this class was generated from the following files:

- artdaq/artdaq/ArtModules/detail/TransferWrapper.hh
- artdaq/artdaq/ArtModules/detail/TransferWrapper.cc

7.96 artdaq::unregister_monitor_ Class Reference

[unregister_monitor_](#) Command class

Inheritance diagram for artdaq::unregister_monitor_:



Public Member Functions

- [unregister_monitor_\(xmlrpc_commander &c\)](#)
unregister_monitor_ Constructor

Additional Inherited Members

7.96.1 Detailed Description

[unregister_monitor_](#) Command class

Definition at line 633 of file `xmlrpc_commander.cc`.

7.96.2 Constructor & Destructor Documentation

7.96.2.1 artdaq::unregister_monitor_::unregister_monitor_(xmlrpc_commander & c) [inline]

[unregister_monitor_](#) Constructor

Parameters

c	xmlrpc_commander to send transition commands to
---	---

Definition at line 640 of file `xmlrpc_commander.cc`.

The documentation for this class was generated from the following file:

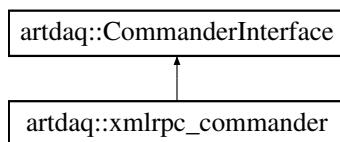
- artdaq/artdaq/ExternalComms/xmlrpc_commander.cc

7.97 artdaq::xmlrpc_commander Class Reference

The [xmlrpc_commander](#) class serves as the XMLRPC server run in each artdaq application.

```
#include <artdaq/ExternalComms/xmlrpc_commander.hh>
```

Inheritance diagram for artdaq::xmlrpc_commander:



Public Member Functions

- `xmlrpc_commander` (`fhicl::ParameterSet ps, artdaq::Commandable &commandable`)
`xmlrpc_commander` Constructor
- `void run_server () override`
Run the XMLRPC server.
- `std::string send_register_monitor (std::string monitor_fhicl) override`
Send a register_monitor command over XMLRPC
- `std::string send_unregister_monitor (std::string monitor_label) override`
Send an unregister_monitor command over XMLRPC

Public Attributes

- `std::mutex mutex_`
XMLRPC mutex.

Additional Inherited Members

7.97.1 Detailed Description

The `xmlrpc_commander` class serves as the XMLRPC server run in each artdaq application.

Definition at line 17 of file `xmlrpc_commander.hh`.

7.97.2 Constructor & Destructor Documentation

7.97.2.1 `artdaq::xmlrpc_commander::xmlrpc_commander (fhicl::ParameterSet ps, artdaq::Commandable & commandable)`

`xmlrpc_commander` Constructor

Parameters

<code>ps</code>	ParameterSet used for configuring <code>xmlrpc_commander</code>
<code>commandable</code>	<code>artdaq::Commandable</code> object to send transition commands to

`xmlrpc_commander` accepts the following Parameters:
`id`: For XMLRPC, the ID should be the port to listen on
`server_url`: When sending, location of XMLRPC server
*

Definition at line 688 of file `xmlrpc_commander.cc`.

7.97.3 Member Function Documentation

7.97.3.1 `std::string artdaq::xmlrpc_commander::send_register_monitor (std::string monitor_fhicl) [override], [virtual]`

Send a register_monitor command over XMLRPC

Parameters

<i>monitor_fhicl</i>	FHiCL string containing monitor configuration
----------------------	---

Returns

Return status from XMLRPC

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 812 of file `xmlrpc_commander.cc`.

7.97.3.2 std::string artdaq::xmlrpc_commander::send_unregister_monitor (std::string *monitor_label*) [override], [virtual]

Send an unregister_monitor command over XMLRPC

Parameters

<i>monitor_label</i>	Label of the monitor to unregister
----------------------	------------------------------------

Returns

Return status from XMLRPC

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 841 of file `xmlrpc_commander.cc`.

The documentation for this class was generated from the following files:

- artdaq/artdaq/ExternalComms/xmlrpc_commander.hh
- artdaq/artdaq/ExternalComms/xmlrpc_commander.cc

Chapter 8

File Documentation

8.1 artdaq/artdaq/DAQdata/TCP_listen_fd.hh File Reference

Functions

- int [TCP_listen_fd](#) (int port, int rcvbuf)

Create a TCP listening socket on the given port and INADDR_ANY, with the given receive buffer.

8.1.1 Detailed Description

Defines a generator function for a TCP listen socket

Definition in file [TCP_listen_fd.hh](#).

8.1.2 Function Documentation

8.1.2.1 int TCP_listen_fd (int port, int rcvbuf)

Create a TCP listening socket on the given port and INADDR_ANY, with the given receive buffer.

Parameters

<i>port</i>	Port to listen on
<i>rcvbuf</i>	Receive buffer for socket. Set to 0 for TCP automatic buffer size

Returns

fd of new socket

Definition at line 21 of file [TCP_listen_fd.cc](#).

8.2 artdaq/artdaq/DAQdata/TCPConnect.hh File Reference

```
#include <netinet/in.h>
```

Functions

- int [ResolveHost](#) (char const *host_in, in_addr &addr)
Convert a string hostname to a in_addr suitable for socket communication.
- int [ResolveHost](#) (char const *host_in, int dfilt_port, sockaddr_in &sin)
Convert a string hostname and port to a sockaddr_in suitable for socket communication.
- int [TCPConnect](#) (char const *host_in, int dfilt_port, long flags=0, int sndbufsiz=0)
Connect to a host on a given port.

8.2.1 Detailed Description

Provides utility functions for connecting TCP sockets

Definition in file [TCPConnect.hh](#).

8.2.2 Function Documentation

8.2.2.1 int ResolveHost (char const * host_in, in_addr & addr)

Convert a string hostname to a in_addr suitable for socket communication.

Parameters

	<i>host_in</i>	Name or IP of host to resolve
out	<i>addr</i>	in_addr object populated with resolved host

Returns

0 if success, -1 if gethostbyname fails

Definition at line 27 of file [TCPConnect.cc](#).

8.2.2.2 int ResolveHost (char const * host_in, int dfilt_port, sockaddr_in & sin)

Convert a string hostname and port to a sockaddr_in suitable for socket communication.

Parameters

	<i>host_in</i>	Name or IP of host to resolve
	<i>dfilt_port</i>	POrt to populate in output
out	<i>sin</i>	sockaddr_in object populated with resolved host and port

Returns

0 if success, -1 if gethostbyname fails

Definition at line 70 of file [TCPConnect.cc](#).

8.2.2.3 int TCPConnect (char const * host_in, int dfilt_port, long flags = 0, int sndbufsiz = 0)

Connect to a host on a given port.

Parameters

<i>host_in</i>	Name or IP of the host to connect to
<i>dflt_port</i>	Port to connect to
<i>flags</i>	TCP flags to use for the socket
<i>sndbufsiz</i>	Size of the send buffer. Set to 0 for automatic send buffer management

Returns

File descriptor of connected socket.

Definition at line 122 of file TCPConnect.cc.

8.3 artdaq/artdaq/DAQrate/quiet_mpi.hh File Reference

```
#include <mpi.h>
```

8.3.1 Detailed Description

Use this header, rather than mpi.h directly, to include the GCC pragma magic to silence warnings about unused variables in mpi.h

Definition in file [quiet_mpi.hh](#).

Index

~CommandableFragmentGenerator
 artdaq::CommandableFragmentGenerator, 77

~DataLoggerCore
 artdaq::DataLoggerCore, 99

~DataReceiverCore
 artdaq::DataReceiverCore, 103

~DispatcherCore
 artdaq::DispatcherCore, 118

~EventBuilderCore
 artdaq::EventBuilderCore, 127

~RoutingMasterCore
 artdaq::RoutingMasterCore, 195

_commandable
 artdaq::CommanderInterface, 92

add_periodic
 Timeout, 241, 243

add_relative
 Timeout, 243

AddFragment
 artdaq::SharedMemoryEventManager, 213

addMonitoredQuantityName
 artdaq::StatisticsHelper, 232

AddReceiverToken
 artdaq::RoutingMasterPolicy, 201

AddRequest
 artdaq::RequestSender, 183

addRequest
 artdaq::detail::RequestMessage, 180

addSample
 artdaq::StatisticsHelper, 232

analyze
 artdaq::EventDump, 130
 artdaq::FragmentSniffer, 137

anonymous_namespace{TransferWrapper.cc}, 21

anonymous_namespace{genToArt.cc}, 19
 process_cmd_line, 19
 process_data, 19

anonymous_namespace{genToArt.cc}::ThrottledGenerator,
 238
 getNext, 239
 numFragIDs, 239
 start, 239
 stop, 240
 ThrottledGenerator, 239

anonymous_namespace{rootOutputConfigurationTools.-
 cc}, 20
 maxCriterionSpecified, 20

anonymous_namespace{shared_memory_reader_t.cc},
 20
 basic_test, 21

applyRequests
 artdaq::CommandableFragmentGenerator, 77

applyRequestsBufferMode
 artdaq::CommandableFragmentGenerator, 77

applyRequestsIgnoredMode
 artdaq::CommandableFragmentGenerator, 77

applyRequestsSingleMode
 artdaq::CommandableFragmentGenerator, 79

applyRequestsWindowMode
 artdaq::CommandableFragmentGenerator, 79

art, 21
 printProcessHistoryID, 22
 printProcessMap, 23
 ReadObjectAny, 23

art::ArtdaqInput
 ArtdaqInput, 39
 hasMoreData, 39
 operator=, 39
 readFile, 39
 readNext, 40

art::ArtdaqInput< U >, 38

art::BinaryFileOutput, 44
 BinaryFileOutput, 44

art::BinaryMPIOutput, 45
 BinaryMPIOutput, 45

art::NetMonWrapper, 164
 NetMonWrapper, 165
 receiveInitMessage, 165
 receiveMessage, 165

art::RootMPIOutput, 186
 RootMPIOutput, 187

art::Source_generator< artdaq::detail::SharedMemory-
 Reader<>>, 228

art::Source_generator< ArtdaqInput< artdaq::Transfer-
 Wrapper >>, 229

art::Source_generator< ArtdaqInput< NetMonWrapper >
 >, 229

art::TransferOutput, 251
 TransferOutput, 251

art::detail::DummyProductCache, 121
 product, 121
art_config_file
 artdaq::art_config_file, 37
ardaq, 23
 cmd_::getParam< art::RunID >, 28
 cmd_::getParam< fhicl::ParameterSet >, 29
 cmd_::getParam< std::string >, 29
 exception_msg, 29, 30
 infoFilename, 30
 makeCommandableFragmentGenerator, 32
 MakeCommanderPlugin, 32
 makeFunc_t, 28
 makeRoutingMasterPolicy, 32
 MakeTransferPlugin, 33
ardaq/artdaq/DAQdata/TCP_listen_fd.hh, 259
ardaq/artdaq/DAQdata/TCPConnect.hh, 259
ardaq/artdaq/DAQrate/quiet_mpi.hh, 261
ardaq::GenericFragmentSimulator
 DEAD_BEEF, 140
 EMPTY, 140
 FRAG_ID, 140
 RANDOM, 140
ardaq::TransferInterface
 kErrorNotRequiringException, 247
 kReceive, 247
 kSend, 247
 kSuccess, 247
 kTimeout, 247
ardaq::detail
 EndOfRun, 34
 Normal, 34
 RouteBySendCount, 35
 RouteBySequenceID, 35
ardaq::AutodetectTransfer, 40
 AutodetectTransfer, 41
 copyFragment, 41
 moveFragment, 41
 receiveFragment, 43
 receiveFragmentData, 43
 receiveFragmentHeader, 43
ardaq::BoardReaderApp, 46
 BoardReaderApp, 47
 BootedEnter, 47
 do_initialize, 47
 do_pause, 48
 do_reinitialize, 48
 do_resume, 48
 do_shutdown, 49
 do_soft_initialize, 49
 do_start, 49
 do_stop, 50
 operator=, 50
 report, 50
ardaq::BoardReaderCore, 51
 BoardReaderCore, 52
 GetDataSenderManagerPtr, 52
 initialize, 52
 operator=, 53
 pause, 53
 process_fragments, 53
 reinitialize, 53
 report, 55
 resume, 55
 shutdown, 55
 soft_initialize, 55
 start, 56
 stop, 56
ardaq::BuildInfo
 beginRun, 58
 BuildInfo, 58
 produce, 59
ardaq::BuildInfo< instanceName, Pkgs >, 57
ardaq::CapacityTestPolicy, 59
 CapacityTestPolicy, 60
 GetCurrentTable, 60
ardaq::Commandable, 64
 badTransition, 66
 BootedEnter, 66
 Commandable, 66
 current_state, 66
 do_initialize, 67
 do_pause, 67
 do_reinitialize, 67
 do_resume, 67
 do_shutdown, 68
 do_soft_initialize, 68
 do_start, 68
 do_stop, 68
 in_run_failure, 69
 InRunExit, 69
 initialize, 69
 legal_commands, 69
 operator=, 70
 pause, 70
 register_monitor, 70
 reinitialize, 70
 report, 71
 resume, 71
 shutdown, 71
 soft_initialize, 71
 start, 72
 status, 72
 stop, 72
 unregister_monitor, 72
ardaq::CommandableFragmentGenerator, 73
 ~CommandableFragmentGenerator, 77
 applyRequests, 77

applyRequestsBufferMode, 77
 applyRequestsIgnoredMode, 77
 applyRequestsSingleMode, 79
 applyRequestsWindowMode, 79
 board_id, 79
 check_stop, 79
 CommandableFragmentGenerator, 76
 dataBufferIsTooLarge, 79
 ev_counter, 79
 ev_counter_inc, 80
 exception, 80
 fragment_id, 80
 fragmentIDs, 80
 getDataBufferStats, 81
 getNext, 81
 metricsReportingInstanceName, 81
 PauseCmd, 81
 printMode_, 82
 ReportCmd, 82
 ResumeCmd, 82
 run_number, 82
 sendEmptyFragment, 82
 sendEmptyFragments, 83
 set_exception, 83
 should_stop, 83
 StartCmd, 83
 StopCmd, 84
 subrun_number, 84
 timeout, 84
 timestamp, 84
 waitForDataBufferReady, 84
 artdaq::CommanderInterface, 87
 _commandable, 92
 CommanderInterface, 88
 operator=, 89
 run_server, 89
 send_init, 89
 send_legal_commands, 89
 send_pause, 90
 send_register_monitor, 90
 send_reinit, 90
 send_report, 90
 send_resume, 90
 send_shutdown, 91
 send_soft_init, 91
 send_start, 91
 send_status, 91
 send_stop, 92
 send_unregister_monitor, 92
 artdaq::CompositeDriver, 93
 CompositeDriver, 93
 artdaq::DataLoggerApp, 94
 DataLoggerApp, 95
 do_initialize, 95
 do_pause, 95
 do_reinitialize, 96
 do_resume, 96
 do_shutdown, 96
 do_soft_initialize, 96
 do_start, 96
 do_stop, 98
 operator=, 98
 report, 98
 artdaq::DataLoggerCore, 99
 ~DataLoggerCore, 99
 DataLoggerCore, 99
 initialize, 100
 operator=, 100
 artdaq::DataReceiverCore, 101
 ~DataReceiverCore, 103
 DataReceiverCore, 102
 initialize, 103
 initializeDataReceiver, 103
 logMessage_, 103
 operator=, 105
 pause, 105
 reinitialize, 105
 report, 105
 resume, 105
 shutdown, 106
 soft_initialize, 106
 start, 106
 stop, 106
 artdaq::DataReceiverManager, 107
 byteCount, 108
 count, 108
 DataReceiverManager, 108
 enabled_sources, 108
 GetReceivedFragmentCount, 108
 getSharedMemoryEventManager, 109
 running_sources, 109
 slotCount, 109
 artdaq::DataSenderManager, 109
 count, 111
 DataSenderManager, 110
 destinationCount, 111
 enabled_destinations, 111
 GetRoutingTableEntryCount, 111
 sendFragment, 111
 slotCount, 112
 artdaq::DispatcherApp, 112
 DispatcherApp, 113
 do_initialize, 113
 do_pause, 114
 do_reinitialize, 114
 do_resume, 114
 do_shutdown, 114
 do_soft_initialize, 115

do_start, 115
do_stop, 115
operator=, 115
register_monitor, 115
report, 117
unregister_monitor, 117
artdaq::DispatcherCore, 117
~DispatcherCore, 118
DispatcherCore, 118
initialize, 119
operator=, 119
register_monitor, 119
unregister_monitor, 121
artdaq::EventBuilderApp, 122
BootedEnter, 123
do_initialize, 123
do_pause, 124
do_reinitialize, 124
do_resume, 124
do_shutdown, 124
do_soft_initialize, 125
do_start, 125
do_stop, 125
EventBuilderApp, 123
operator=, 125
report, 126
artdaq::EventBuilderCore, 126
~EventBuilderCore, 127
EventBuilderCore, 127
initialize, 127
operator=, 128
artdaq::EventDump, 128
analyze, 130
EventDump, 129
artdaq::FragmentReceiverManager, 132
byteCount, 134
count, 134
enabled_sources, 134
FragmentReceiverManager, 133
recvFragment, 134
slotCount, 134
artdaq::FragmentSniffer, 135
analyze, 137
FragmentSniffer, 136
artdaq::FragmentStoreElement, 137
emplace_back, 138
emplace_front, 138
empty, 138
front, 138
GetEndOfData, 138
SetEndOfData, 139
artdaq::GenericFragmentSimulator, 139
content_selector_t, 140
fragmentIDs, 141
GenericFragmentSimulator, 140
getNext, 141
artdaq::GetPackageBuildInfo, 142
getPackageBuildInfo, 142
artdaq::Globals, 142
seedAndRandom_, 143
timevalAsDouble, 143
artdaq::MPISentry, 149
MPISentry, 149, 150
procs, 150
rank, 150
threading_level, 150
artdaq::MPITransfer, 151
copyFragment, 152
MPITransfer, 152
moveFragment, 152
operator=, 152
receiveFragmentData, 153
receiveFragmentHeader, 153
artdaq::MulticastTransfer, 155
copyFragment, 157
moveFragment, 158
MulticastTransfer, 156
receiveFragment, 158
receiveFragmentData, 158
receiveFragmentHeader, 159
artdaq::NetMonHeader, 159
artdaq::NoOpPolicy, 165
GetCurrentTable, 166
NoOpPolicy, 166
artdaq::NullTransfer, 166
copyFragment, 168
moveFragment, 168
NullTransfer, 167
receiveFragment, 168
receiveFragmentData, 168
receiveFragmentHeader, 168
artdaq::PrintBuildInfo, 171
beginRun, 172
PrintBuildInfo, 172
artdaq::RTIDDS, 206
copyFragmentToDDS_, 207
moveFragmentToDDS_, 208
RTIDDS, 207
artdaq::RTIDDS::OctetsListener, 169
on_data_available, 169
receiveFragmentFromDDS, 170
artdaq::RTIDDSTransfer, 208
copyFragment, 209
moveFragment, 209
RTIDDSTransfer, 209
receiveFragment, 210
artdaq::RandomDelayFilter, 173
filter, 174

operator=, 174
 RandomDelayFilter, 173
 reconfigure, 174
 artdaq::RequestSender, 182
 AddRequest, 183
 GetRequestMethod, 184
 operator=, 184
 RemoveRequest, 184
 RequestSender, 183
 SendRequest, 184
 SendRoutingToken, 184
 SetRequestMethod, 185
 artdaq::RoundRobinPolicy, 187
 GetCurrentTable, 188
 RoundRobinPolicy, 188
 artdaq::RoutingMasterApp, 189
 BootedEnter, 190
 do_initialize, 190
 do_pause, 191
 do_reinitialize, 191
 do_resume, 191
 do_shutdown, 192
 do_soft_initialize, 192
 do_start, 192
 do_stop, 193
 operator=, 193
 report, 193
 RoutingMasterApp, 190
 artdaq::RoutingMasterCore, 194
 ~RoutingMasterCore, 195
 initialize, 195
 operator=, 196
 pause, 196
 process_event_table, 196
 reinitialize, 196
 report, 197
 resume, 197
 RoutingMasterCore, 195
 send_event_table, 197
 shutdown, 197
 soft_initialize, 197
 start, 199
 stop, 199
 artdaq::RoutingMasterPolicy, 199
 AddReceiverToken, 201
 GetCurrentTable, 201
 GetMaxNumberOfTokens, 201
 GetReceiverCount, 201
 RoutingMasterPolicy, 200
 artdaq::SharedMemoryEventManager, 210
 AddFragment, 213
 DoneWritingFragment, 213
 endOfData, 213
 endRun, 213
 endSubrun, 214
 GetArtEventCount, 214
 GetBroadcastKey, 214
 GetDroppedDataAddress, 214
 GetFragmentCount, 214
 GetInactiveEventCount, 215
 GetIncompleteEventCount, 215
 GetLockedBufferCount, 215
 GetPendingEventCount, 215
 ReconfigureArt, 215
 runID, 216
 setOverwrite, 216
 setRequestMethod, 216
 SharedMemoryEventManager, 212
 ShutdownArtProcesses, 216
 StartArtProcess, 216
 startRun, 217
 subrunID, 217
 WriteFragmentHeader, 217
 artdaq::ShmemTransfer, 221
 copyFragment, 222
 moveFragment, 222
 receiveFragment, 223
 receiveFragmentData, 223
 receiveFragmentHeader, 223
 ShmemTransfer, 222
 artdaq::StatisticsHelper, 231
 addMonitoredQuantityName, 232
 addSample, 232
 createCollectors, 232
 operator=, 233
 readyToReport, 233
 statsRollingWindowHasMoved, 233
 artdaq::TCPSocketTransfer, 236
 copyFragment, 237
 moveFragment, 237
 receiveFragmentData, 237
 receiveFragmentHeader, 238
 TCPSocketTransfer, 236
 artdaq::TransferInterface, 246
 copyFragment, 248
 CopyStatus, 247
 destination_rank, 248
 moveFragment, 248
 operator=, 249
 receiveFragment, 249
 receiveFragmentData, 249
 receiveFragmentHeader, 250
 Role, 247
 role, 250
 source_rank, 250
 TransferInterface, 248
 uniqueLabel, 250
 artdaq::TransferTest, 252

runTest, 253
TransferTest, 252
artdaq::TransferWrapper, 253
receiveInitMessage, 254
receiveMessage, 254
TransferWrapper, 253
artdaq::art_config_file, 37
art_config_file, 37
getFileName, 37
artdaq::cmd_, 60
cmd_, 62
execute, 62
execute_, 62
getParam, 63, 64
artdaq::detail, 33
operator<<, 35
RequestMessageMode, 34
RoutingMasterMode, 34
artdaq::detail::FragCounter, 130
count, 131
incSlot, 131
minCount, 131
nSlots, 131
setSlot, 132
slotCount, 132
artdaq::detail::RequestHeader, 178
header, 179
isValid, 179
artdaq::detail::RequestMessage, 179
addRequest, 180
buffer, 180
header, 180
size, 180
artdaq::detail::RequestPacket, 181
header, 182
isValid, 182
RequestPacket, 181
artdaq::detail::RoutingAckPacket, 188
artdaq::detail::RoutingPacketEntry, 203
RoutingPacketEntry, 204
artdaq::detail::RoutingPacketHeader, 205
RoutingPacketHeader, 205
artdaq::detail::RoutingToken, 206
artdaq::detail::SharedMemoryReader
hasMoreData, 220
operator=, 220
readFile, 220
readNext, 220
SharedMemoryReader, 219
artdaq::detail::SharedMemoryReader< getDefaultTypes
>, 218
artdaq::init_, 143
defaultTimeout, 144
defaultTimestamp, 144
init_, 144
artdaq::legal_commands_, 145
legal_commands_, 145
artdaq::pause_, 170
defaultTimeout, 171
defaultTimestamp, 171
pause_, 171
artdaq::register_monitor_, 175
register_monitor_, 176
artdaq::reinit_, 176
defaultTimeout, 177
defaultTimestamp, 177
reinit_, 177
artdaq::report_, 177
report_, 178
artdaq::resume_, 185
defaultTimeout, 186
defaultTimestamp, 186
resume_, 186
artdaq::shutdown_, 226
defaultTimeout, 227
shutdown_, 227
artdaq::soft_init_, 227
defaultTimeout, 228
defaultTimestamp, 228
soft_init_, 228
artdaq::start_, 230
defaultTimeout, 231
defaultTimestamp, 231
start_, 230
artdaq::status_, 234
status_, 234
artdaq::stop_, 234
defaultTimeout, 235
defaultTimestamp, 235
stop_, 235
artdaq::unregister_monitor_, 254
unregister_monitor_, 255
artdaq::xmlrpc_commander, 255
send_register_monitor, 256
send_unregister_monitor, 257
xmlrpc_commander, 256
ArtdaqInput
art::ArtdaqInput, 39
artdaqtest::CommandableFragmentGeneratorTest, 85
checkHWStatus_, 86
getNext_, 86
setFireCount, 86
waitForFrags, 87
AutodetectTransfer
artdaq::AutodetectTransfer, 41
badTransition
artdaq::Commandable, 66

basic_test
 anonymous_namespace{shared_memory_reader_t.cc}, 21

beginRun
 artdaq::BuildInfo, 58
 artdaq::PrintBuildInfo, 172

BinaryFileOutput
 art::BinaryFileOutput, 44

BinaryMPIOutput
 art::BinaryMPIOutput, 45

board_id
 artdaq::CommandableFragmentGenerator, 79

BoardReaderApp
 artdaq::BoardReaderApp, 47

BoardReaderCore
 artdaq::BoardReaderCore, 52

BootedEnter
 artdaq::BoardReaderApp, 47
 artdaq::Commandable, 66
 artdaq::EventBuilderApp, 123
 artdaq::RoutingMasterApp, 190

buffer
 artdaq::detail::RequestMessage, 180

BuildInfo
 artdaq::BuildInfo, 58

Builder, 56
 Builder, 57

byteCount
 artdaq::DataReceiverManager, 108
 artdaq::FragmentReceiverManager, 134

cancel_timeout
 Timeout, 243

CapacityTestPolicy
 artdaq::CapacityTestPolicy, 60

check_stop
 artdaq::CommandableFragmentGenerator, 79

checkHWStatus_
 artdaqtest::CommandableFragmentGeneratorTest, 86

cmd_
 artdaq::cmd_, 62

cmd_::getParam< art::RunID >
 artdaq, 28

cmd_::getParam< fhicl::ParameterSet >
 artdaq, 29

cmd_::getParam< std::string >
 artdaq, 29

Commandable
 artdaq::Commandable, 66

CommandableFragmentGenerator
 artdaq::CommandableFragmentGenerator, 76

CommanderInterface
 artdaq::CommanderInterface, 88

CompositeDriver
 artdaq::CompositeDriver, 93

connect
 NetMonTransportService, 161
 NetMonTransportServiceInterface, 163

content_selector_t
 artdaq::GenericFragmentSimulator, 140

copy_in_timeout
 Timeout, 244

copyFragment
 artdaq::AutodetectTransfer, 41
 artdaq::MPITransfer, 152
 artdaq::MulticastTransfer, 157
 artdaq::NullTransfer, 168
 artdaq::RTIDDSTransfer, 209
 artdaq::ShmemTransfer, 222
 artdaq::TCPSocketTransfer, 237
 artdaq::TransferInterface, 248

copyFragmentToDDS_
 artdaq::RTIDDS, 207

CopyStatus
 artdaq::TransferInterface, 247

count
 artdaq::DataReceiverManager, 108
 artdaq::DataSenderManager, 111
 artdaq::detail::FragCounter, 131
 artdaq::FragmentReceiverManager, 134

createCollectors
 artdaq::StatisticsHelper, 232

current_state
 artdaq::Commandable, 66

DEAD_BEEF
 artdaq::GenericFragmentSimulator, 140

daqrate.Re, 175

dataBufferIsTooLarge
 artdaq::CommandableFragmentGenerator, 79

DataLoggerApp
 artdaq::DataLoggerApp, 95

DataLoggerCore
 artdaq::DataLoggerCore, 99

DataReceiverCore
 artdaq::DataReceiverCore, 102

dataReceiverCount
 NetMonTransportService, 161

DataReceiverManager
 artdaq::DataReceiverManager, 108

DataSenderManager
 artdaq::DataSenderManager, 110

defaultTimeout
 artdaq::init_, 144
 artdaq::pause_, 171
 artdaq::reinit_, 177
 artdaq::resume_, 186

artdaq::shutdown_, 227
artdaq::soft_init_, 228
artdaq::start_, 231
artdaq::stop_, 235
defaultTimestamp
 artdaq::init_, 144
 artdaq::pause_, 171
 artdaq::reinit_, 177
 artdaq::resume_, 186
 artdaq::soft_init_, 228
 artdaq::start_, 231
 artdaq::stop_, 235
destination_rank
 artdaq::TransferInterface, 248
destinationCount
 artdaq::DataSenderManager, 111
disconnect
 NetMonTransportService, 161
 NetMonTransportServiceInterface, 163
DispatcherApp
 artdaq::DispatcherApp, 113
DispatcherCore
 artdaq::DispatcherCore, 118
do_initialize
 artdaq::BoardReaderApp, 47
 artdaq::Commandable, 67
 artdaq::DataLoggerApp, 95
 artdaq::DispatcherApp, 113
 artdaq::EventBuilderApp, 123
 artdaq::RoutingMasterApp, 190
do_pause
 artdaq::BoardReaderApp, 48
 artdaq::Commandable, 67
 artdaq::DataLoggerApp, 95
 artdaq::DispatcherApp, 114
 artdaq::EventBuilderApp, 124
 artdaq::RoutingMasterApp, 191
do_reinitialize
 artdaq::BoardReaderApp, 48
 artdaq::Commandable, 67
 artdaq::DataLoggerApp, 96
 artdaq::DispatcherApp, 114
 artdaq::EventBuilderApp, 124
 artdaq::RoutingMasterApp, 191
do_resume
 artdaq::BoardReaderApp, 48
 artdaq::Commandable, 67
 artdaq::DataLoggerApp, 96
 artdaq::DispatcherApp, 114
 artdaq::EventBuilderApp, 124
 artdaq::RoutingMasterApp, 191
do_shutdown
 artdaq::BoardReaderApp, 49
 artdaq::Commandable, 68
 artdaq::DataLoggerApp, 96
 artdaq::DispatcherApp, 114
 artdaq::EventBuilderApp, 124
 artdaq::RoutingMasterApp, 192
do_soft_initialize
 artdaq::BoardReaderApp, 49
 artdaq::Commandable, 68
 artdaq::DataLoggerApp, 96
 artdaq::DispatcherApp, 115
 artdaq::EventBuilderApp, 125
 artdaq::RoutingMasterApp, 192
do_start
 artdaq::BoardReaderApp, 49
 artdaq::Commandable, 68
 artdaq::DataLoggerApp, 96
 artdaq::DispatcherApp, 115
 artdaq::EventBuilderApp, 125
 artdaq::RoutingMasterApp, 192
do_stop
 artdaq::BoardReaderApp, 50
 artdaq::Commandable, 68
 artdaq::DataLoggerApp, 98
 artdaq::DispatcherApp, 115
 artdaq::EventBuilderApp, 125
 artdaq::RoutingMasterApp, 193
DoneWritingFragment
 artdaq::SharedMemoryEventManager, 213

EMPTY
 artdaq::GenericFragmentSimulator, 140
emplace_back
 artdaq::FragmentStoreElement, 138
emplace_front
 artdaq::FragmentStoreElement, 138
empty
 artdaq::FragmentStoreElement, 138
enabled_destinations
 artdaq::DataSenderManager, 111
enabled_sources
 artdaq::DataReceiverManager, 108
 artdaq::FragmentReceiverManager, 134
EndOfRun
 artdaq::detail, 34
endOfData
 artdaq::SharedMemoryEventManager, 213
endRun
 artdaq::SharedMemoryEventManager, 213
endSubrun
 artdaq::SharedMemoryEventManager, 214
ev_counter
 artdaq::CommandableFragmentGenerator, 79
ev_counter_inc
 artdaq::CommandableFragmentGenerator, 80
EventBuilderApp

artdaq::EventBuilderApp, 123
 EventBuilderCore
 artdaq::EventBuilderCore, 127
 EventDump
 artdaq::EventDump, 129
 exception
 artdaq::CommandableFragmentGenerator, 80
 exception_msg
 artdaq, 29, 30
 execute
 artdaq::cmd_, 62
 execute_
 artdaq::cmd_, 62

FRAG_ID
 artdaq::GenericFragmentSimulator, 140
 fake_single_module_process
 MPRGlobalTestFixture, 154
 fake_single_process_branch
 MPRGlobalTestFixture, 155
 filter
 artdaq::RandomDelayFilter, 174
 fragment_id
 artdaq::CommandableFragmentGenerator, 80
 fragmentIDs
 artdaq::CommandableFragmentGenerator, 80
 artdaq::GenericFragmentSimulator, 141
 FragmentReceiverManager
 artdaq::FragmentReceiverManager, 133
 FragmentSniffer
 artdaq::FragmentSniffer, 136
 front
 artdaq::FragmentStoreElement, 138

GenericFragmentSimulator
 artdaq::GenericFragmentSimulator, 140
 get_next_expired_timeout
 Timeout, 244
 get_next_timeout_delay
 Timeout, 244
 get_next_timeout_msdl
 Timeout, 244
 GetArtEventCount
 artdaq::SharedMemoryEventManager, 214
 GetBroadcastKey
 artdaq::SharedMemoryEventManager, 214
 getBroadcastKey
 ShmRTestFixture, 225
 GetCurrentTable
 artdaq::CapacityTestPolicy, 60
 artdaq::NoOpPolicy, 166
 artdaq::RoundRobinPolicy, 188
 artdaq::RoutingMasterPolicy, 201
 getDataBufferStats
 artdaq::CommandableFragmentGenerator, 81

 GetDataSenderManagerPtr
 artdaq::BoardReaderCore, 52
 GetDroppedDataAddress
 artdaq::SharedMemoryEventManager, 214
 GetEndOfData
 artdaq::FragmentStoreElement, 138
 getFileName
 artdaq::art_config_file, 37
 GetFragmentCount
 artdaq::SharedMemoryEventManager, 214
 GetInactiveEventCount
 artdaq::SharedMemoryEventManager, 215
 GetIncompleteEventCount
 artdaq::SharedMemoryEventManager, 215
 getKey
 ShmRTestFixture, 225
 GetLockedBufferCount
 artdaq::SharedMemoryEventManager, 215
 GetMaxNumberOfTokens
 artdaq::RoutingMasterPolicy, 201
 getNext
 anonymous_namespace{genToArt.cc}::Throttled-
 Generator, 239
 artdaq::CommandableFragmentGenerator, 81
 artdaq::GenericFragmentSimulator, 141
 getNext_
 artdaqtest::CommandableFragmentGeneratorTest,
 86
 getPackageBuildInfo
 artdaq::GetPackageBuildInfo, 142
 getParam
 artdaq::cmd_, 63, 64
 GetPendingEventCount
 artdaq::SharedMemoryEventManager, 215
 getPset
 RoutingMasterTest, 203
 GetReceivedFragmentCount
 artdaq::DataReceiverManager, 108
 GetReceiverCount
 artdaq::RoutingMasterPolicy, 201
 GetRequestMethod
 artdaq::RequestSender, 184
 GetRoutingTableEntryCount
 artdaq::DataSenderManager, 111
 getSharedMemoryEventManager
 artdaq::DataReceiverManager, 109
 gf
 ShmRTestFixture, 225

 hasMoreData
 art::ArtdaqInput, 39
 artdaq::detail::SharedMemoryReader, 220
 header
 artdaq::detail::RequestHeader, 179

artdaq::detail::RequestMessage, 180
artdaq::detail::RequestPacket, 182
helper
 ShmRTestFixture, 225

in_run_failure
 artdaq::Commandable, 69
InRunExit
 artdaq::Commandable, 69
incSlot
 artdaq::detail::FragCounter, 131
infoFilename
 artdaq, 30
init_
 artdaq::init_, 144
initialize
 artdaq::BoardReaderCore, 52
 artdaq::Commandable, 69
 artdaq::DataLoggerCore, 100
 artdaq::DataReceiverCore, 103
 artdaq::DispatcherCore, 119
 artdaq::EventBuilderCore, 127
 artdaq::RoutingMasterCore, 195
initializeDataReceiver
 artdaq::DataReceiverCore, 103
is_consistent
 Timeout, 245
IsLocked
 LockFile, 146
isValid
 artdaq::detail::RequestHeader, 179
 artdaq::detail::RequestPacket, 182

kErrorNotRequiringException
 artdaq::TransferInterface, 247
kReceive
 artdaq::TransferInterface, 247
kSend
 artdaq::TransferInterface, 247
kSuccess
 artdaq::TransferInterface, 247
kTimeout
 artdaq::TransferInterface, 247

legal_commands
 artdaq::Commandable, 69
legal_commands_
 artdaq::legal_commands_, 145
listen
 NetMonTransportServiceInterface, 163
LockFile, 145
 IsLocked, 146
 LockFile, 146
 LockFile, 146
logMessage_

 artdaq::DataReceiverCore, 103
MPIProg, 148
 MPIProg, 148
 MPIProg, 148
MPISentry
 artdaq::MPISentry, 149, 150
MPITransfer
 artdaq::MPITransfer, 152
MPRGlobaTestFixture, 153
 fake_single_module_process, 154
 fake_single_process_branch, 155
makeCommandableFragmentGenerator
 artdaq, 32
MakeCommanderPlugin
 artdaq, 32
makeFunc_t
 artdaq, 28
makeRoutingMasterPolicy
 artdaq, 32
MakeTransferPlugin
 artdaq, 33
maxCriterionSpecified
 anonymous_namespace{rootOutputConfiguration-
 Tools.cc}, 20
MessHead, 147
 MessType, 147
MessType
 MessHead, 147
metricsReportingInstanceName
 artdaq::CommandableFragmentGenerator, 81
minCount
 artdaq::detail::FragCounter, 131
moveFragment
 artdaq::AutodetectTransfer, 41
 artdaq::MPITransfer, 152
 artdaq::MulticastTransfer, 158
 artdaq::NullTransfer, 168
 artdaq::RTIDDSTransfer, 209
 artdaq::ShmemTransfer, 222
 artdaq::TCPSocketTransfer, 237
 artdaq::TransferInterface, 248
moveFragmentToDDS_
 artdaq::RTIDDS, 208
MulticastTransfer
 artdaq::MulticastTransfer, 156

nSlots
 artdaq::detail::FragCounter, 131
NetMonTransportService, 160
 connect, 161
 dataReceiverCount, 161
 disconnect, 161
 NetMonTransportService, 160
 NetMonTransportService, 160

receiveInitMessage, 161
 receiveMessage, 161
 sendMessage, 162
NetMonTransportServiceInterface, 162
 connect, 163
 disconnect, 163
 listen, 163
 receiveInitMessage, 163
 receiveMessage, 163
 sendMessage, 164
NetMonWrapper
 art::NetMonWrapper, 165
NoOpPolicy
 artdaq::NoOpPolicy, 166
Normal
 artdaq::detail, 34
NullTransfer
 artdaq::NullTransfer, 167
numFragIDs
 anonymous_namespace{genToArt.cc}::Throttled-Generator, 239

on_data_available
 artdaq::RTIDDS::OctetsListener, 169
operator<<
 artdaq::detail, 35
operator=
 art::ArtdaqInput, 39
 artdaq::BoardReaderApp, 50
 artdaq::BoardReaderCore, 53
 artdaq::Commandable, 70
 artdaq::CommanderInterface, 89
 artdaq::DataLoggerApp, 98
 artdaq::DataLoggerCore, 100
 artdaq::DataReceiverCore, 105
 artdaq::detail::SharedMemoryReader, 220
 artdaq::DispatcherApp, 115
 artdaq::DispatcherCore, 119
 artdaq::EventBuilderApp, 125
 artdaq::EventBuilderCore, 128
 artdaq::MPITransfer, 152
 artdaq::RandomDelayFilter, 174
 artdaq::RequestSender, 184
 artdaq::RoutingMasterApp, 193
 artdaq::RoutingMasterCore, 196
 artdaq::StatisticsHelper, 233
 artdaq::TransferInterface, 249

pause
 artdaq::BoardReaderCore, 53
 artdaq::Commandable, 70
 artdaq::DataReceiverCore, 105
 artdaq::RoutingMasterCore, 196
pause_
 artdaq::pause_, 171

 PauseCmd
 artdaq::CommandableFragmentGenerator, 81
PrintBuildInfo
 artdaq::PrintBuildInfo, 172
printMode_
 artdaq::CommandableFragmentGenerator, 82
printProcessHistoryID
 art, 22
printProcessMap
 art, 23
process_cmd_line
 anonymous_namespace{genToArt.cc}, 19
process_data
 anonymous_namespace{genToArt.cc}, 19
process_event_table
 artdaq::RoutingMasterCore, 196
process.fragments
 artdaq::BoardReaderCore, 53
procs
 artdaq::MPISentry, 150
produce
 artdaq::BuildInfo, 59
product
 art::detail::DummyProductCache, 121

RANDOM
 artdaq::GenericFragmentSimulator, 140
RTIDDS
 artdaq::RTIDDS, 207
RTIDDSTransfer
 artdaq::RTIDDSTransfer, 209
RandomDelayFilter
 artdaq::RandomDelayFilter, 173
rank
 artdaq::MPISentry, 150
readFile
 art::ArtdaqInput, 39
 artdaq::detail::SharedMemoryReader, 220
readNext
 art::ArtdaqInput, 40
 artdaq::detail::SharedMemoryReader, 220
ReadObjectAny
 art, 23
reader
 ShmRTestFixture, 225
readyToReport
 artdaq::StatisticsHelper, 233
receiveFragment
 artdaq::AutodetectTransfer, 43
 artdaq::MulticastTransfer, 158
 artdaq::NullTransfer, 168
 artdaq::RTIDDSTransfer, 210
 artdaq::ShmemTransfer, 223
 artdaq::TransferInterface, 249

receiveFragmentData
 artdaq::AutodetectTransfer, 43
 artdaq::MPITransfer, 153
 artdaq::MulticastTransfer, 158
 artdaq::NullTransfer, 168
 artdaq::ShmemTransfer, 223
 artdaq::TCPSocketTransfer, 237
 artdaq::TransferInterface, 249

receiveFragmentFromDDS
 artdaq::RTIDDS::OctetsListener, 170

receiveFragmentHeader
 artdaq::AutodetectTransfer, 43
 artdaq::MPITransfer, 153
 artdaq::MulticastTransfer, 159
 artdaq::NullTransfer, 168
 artdaq::ShmemTransfer, 223
 artdaq::TCPSocketTransfer, 238
 artdaq::TransferInterface, 250

receiveInitMessage
 art::NetMonWrapper, 165
 artdaq::TransferWrapper, 254
 NetMonTransportService, 161
 NetMonTransportServiceInterface, 163

receiveMessage
 art::NetMonWrapper, 165
 artdaq::TransferWrapper, 254
 NetMonTransportService, 161
 NetMonTransportServiceInterface, 163

reconfigure
 artdaq::RandomDelayFilter, 174

ReconfigureArt
 artdaq::SharedMemoryEventManager, 215

recvFragment
 artdaq::FragmentReceiverManager, 134

register_monitor
 artdaq::Commandable, 70
 artdaq::DispatcherApp, 115
 artdaq::DispatcherCore, 119

register_monitor_
 artdaq::register_monitor_, 176

reinit_
 artdaq::reinit_, 177

reinitialize
 artdaq::BoardReaderCore, 53
 artdaq::Commandable, 70
 artdaq::DataReceiverCore, 105
 artdaq::RoutingMasterCore, 196

RemoveRequest
 artdaq::RequestSender, 184

report
 artdaq::BoardReaderApp, 50
 artdaq::BoardReaderCore, 55
 artdaq::Commandable, 71
 artdaq::DataLoggerApp, 98

 artdaq::DataReceiverCore, 105
 artdaq::DispatcherApp, 117
 artdaq::EventBuilderApp, 126
 artdaq::RoutingMasterApp, 193
 artdaq::RoutingMasterCore, 197

report_
 artdaq::report_, 178

ReportCmd
 artdaq::CommandableFragmentGenerator, 82

RequestMessageMode
 artdaq::detail, 34

RequestPacket
 artdaq::detail::RequestPacket, 181

RequestSender
 artdaq::RequestSender, 183

ResolveHost
 TCPConnect.hh, 260

resume
 artdaq::BoardReaderCore, 55
 artdaq::Commandable, 71
 artdaq::DataReceiverCore, 105
 artdaq::RoutingMasterCore, 197

resume_
 artdaq::resume_, 186

ResumeCmd
 artdaq::CommandableFragmentGenerator, 82

Role
 artdaq::TransferInterface, 247

role
 artdaq::TransferInterface, 250

RootMPIOutput
 art::RootMPIOutput, 187

RoundRobinPolicy
 artdaq::RoundRobinPolicy, 188

RouteBySendCount
 artdaq::detail, 35

RouteBySequenceID
 artdaq::detail, 35

RoutingMasterApp
 artdaq::RoutingMasterApp, 190

RoutingMasterCore
 artdaq::RoutingMasterCore, 195

RoutingMasterMode
 artdaq::detail, 34

RoutingMasterPolicy
 artdaq::RoutingMasterPolicy, 200

RoutingMasterTest, 201
 getPset, 203
 RoutingMasterTest, 202
 RoutingMasterTest, 202

RoutingPacketEntry
 artdaq::detail::RoutingPacketEntry, 204

RoutingPacketHeader
 artdaq::detail::RoutingPacketHeader, 205

run_number
 artdaq::CommandableFragmentGenerator, 82

run_server
 artdaq::CommanderInterface, 89

runID
 artdaq::SharedMemoryEventManager, 216

runTest
 artdaq::TransferTest, 253

running_sources
 artdaq::DataReceiverManager, 109

seedAndRandom_
 artdaq::Globals, 143

send_event_table
 artdaq::RoutingMasterCore, 197

send_init
 artdaq::CommanderInterface, 89

send_legal_commands
 artdaq::CommanderInterface, 89

send_pause
 artdaq::CommanderInterface, 90

send_register_monitor
 artdaq::CommanderInterface, 90
 artdaq::xmlrpc_commander, 256

send_reinit
 artdaq::CommanderInterface, 90

send_report
 artdaq::CommanderInterface, 90

send_resume
 artdaq::CommanderInterface, 90

send_shutdown
 artdaq::CommanderInterface, 91

send_soft_init
 artdaq::CommanderInterface, 91

send_start
 artdaq::CommanderInterface, 91

send_status
 artdaq::CommanderInterface, 91

send_stop
 artdaq::CommanderInterface, 92

send_unregister_monitor
 artdaq::CommanderInterface, 92
 artdaq::xmlrpc_commander, 257

sendEmptyFragment
 artdaq::CommandableFragmentGenerator, 82

sendEmptyFragments
 artdaq::CommandableFragmentGenerator, 83

sendFragment
 artdaq::DataSenderManager, 111

sendMessage
 NetMonTransportService, 162
 NetMonTransportServiceInterface, 164

SendRequest
 artdaq::RequestSender, 184

SendRoutingToken
 artdaq::RequestSender, 184

set_exception
 artdaq::CommandableFragmentGenerator, 83

SetEndOfData
 artdaq::FragmentStoreElement, 139

setFireCount
 artdaqtest::CommandableFragmentGeneratorTest, 86

setOverwrite
 artdaq::SharedMemoryEventManager, 216

SetRequestMode
 artdaq::RequestSender, 185

setRequestMode
 artdaq::SharedMemoryEventManager, 216

setSlot
 artdaq::detail::FragCounter, 132

SharedMemoryEventManager
 artdaq::SharedMemoryEventManager, 212

SharedMemoryReader
 artdaq::detail::SharedMemoryReader, 219

ShmRTestFixture, 224
 getBroadcastKey, 225
 getKey, 225
 gf, 225
 helper, 225
 reader, 225
 source_helper, 225
 writer, 226

ShmemTransfer
 artdaq::ShmemTransfer, 222

should_stop
 artdaq::CommandableFragmentGenerator, 83

shutdown
 artdaq::BoardReaderCore, 55
 artdaq::Commandable, 71
 artdaq::DataReceiverCore, 106
 artdaq::RoutingMasterCore, 197

shutdown_
 artdaq::shutdown_, 227

ShutdownArtProcesses
 artdaq::SharedMemoryEventManager, 216

size
 artdaq::detail::RequestMessage, 180

slotCount
 artdaq::DataReceiverManager, 109
 artdaq::DataSenderManager, 112
 artdaq::detail::FragCounter, 132
 artdaq::FragmentReceiverManager, 134

soft_init_
 artdaq::soft_init_, 228

soft_initialize
 artdaq::BoardReaderCore, 55
 artdaq::Commandable, 71

artdaq::DataReceiverCore, 106
artdaq::RoutingMasterCore, 197
source_helper
 ShmRTestFixture, 225
source_rank
 artdaq::TransferInterface, 250
start
 anonymous_namespace{genToArt.cc}::Throttled-
 Generator, 239
 artdaq::BoardReaderCore, 56
 artdaq::Commandable, 72
 artdaq::DataReceiverCore, 106
 artdaq::RoutingMasterCore, 199
start_
 artdaq::start_, 230
StartArtProcess
 artdaq::SharedMemoryEventManager, 216
StartCmd
 artdaq::CommandableFragmentGenerator, 83
startRun
 artdaq::SharedMemoryEventManager, 217
statsRollingWindowHasMoved
 artdaq::StatisticsHelper, 233
status
 artdaq::Commandable, 72
status_
 artdaq::status_, 234
stop
 anonymous_namespace{genToArt.cc}::Throttled-
 Generator, 240
 artdaq::BoardReaderCore, 56
 artdaq::Commandable, 72
 artdaq::DataReceiverCore, 106
 artdaq::RoutingMasterCore, 199
stop_
 artdaq::stop_, 235
StopCmd
 artdaq::CommandableFragmentGenerator, 84
subrun_number
 artdaq::CommandableFragmentGenerator, 84
subrunID
 artdaq::SharedMemoryEventManager, 217

TCP_listen_fd
 TCP_listen_fd.hh, 259
TCP_listen_fd.hh
 TCP_listen_fd, 259
TCPConnect
 TCPConnect.hh, 260
TCPConnect.hh
 ResolveHost, 260
 TCPConnect, 260
TCPSocketTransfer
 artdaq::TCPSocketTransfer, 236

threading_level
 artdaq::MPISentry, 150
ThrottledGenerator
 anonymous_namespace{genToArt.cc}::Throttled-
 Generator, 239
Timeout, 240
 add_periodic, 241, 243
 add_relative, 243
 cancel_timeout, 243
 copy_in_timeout, 244
 get_next_expired_timeout, 244
 get_next_timeout_delay, 244
 get_next_timeout_msdl, 244
 is_consistent, 245
 Timeout, 241
timeout
 artdaq::CommandableFragmentGenerator, 84
Timeout::timeoutspec, 245
timestamp
 artdaq::CommandableFragmentGenerator, 84
timevalAsDouble
 artdaq::Globals, 143
TransferInterface
 artdaq::TransferInterface, 248
TransferOutput
 art::TransferOutput, 251
TransferTest
 artdaq::TransferTest, 252
TransferWrapper
 artdaq::TransferWrapper, 253

uniqueLabel
 artdaq::TransferInterface, 250
unregister_monitor
 artdaq::Commandable, 72
 artdaq::DispatcherApp, 117
 artdaq::DispatcherCore, 121
unregister_monitor_
 artdaq::unregister_monitor_, 255

waitForDataBufferReady
 artdaq::CommandableFragmentGenerator, 84
waitForFrags
 artdaqtest::CommandableFragmentGeneratorTest,
 87
WriteFragmentHeader
 artdaq::SharedMemoryEventManager, 217
writer
 ShmRTestFixture, 226

xmlrpc_commander
 artdaq::xmlrpc_commander, 256